MASTER EN SCIENCES INFORMATIQUES

Etude des performances des méthodes de Load balancing dans les réseaux IP

Materne, Pierre

Award date:
2001

FUNDP
Institut d'Informatique

Rue Grandgagnage, 21
5000 - NAMUR
BELGIQUE

Etude des performances des méthodes de Load Balancing dans les réseaux IP

Mémoire présenté pour l'obtention
du grade de Licencié en Informatique
par

Promoteur: Olivier BONAVENTURE

Pierre MATERNE

Année Académique 2000-2001

Résumé

Le LOAD BALANCING a pour objectif essentiel d'éviter les points de congestion dans un réseau IP en répartissant le plus équitablement possible, sur plusieurs chemins, le trafic devant être acheminé entre deux points quelconques du réseau. Dans cette étude, nous nous intéressons à la répartition du trafic au niveau d'*un* routeur, c'est-à-dire sur les différentes lignes de sortie d'*un* routeur particulier. Nous étudions les performances de plusieurs méthodes de LOAD BALANCING sur du trafic issu d'un réseau IP *réel*. Nous obtenons de bons résultats, pour la plupart des méthodes testées, en ce qui concerne la répartition *globale* du trafic sur les différentes lignes de sortie. L'utilisation de ces méthodes pour effectuer du LOAD BALANCING s'avère donc potentiellement intéressante, en tous cas au niveau d'un routeur pris en particulier. Les résultats obtenus sont nettement moins "bons" en ce qui concerne la répartition *instantanée* du trafic sur chacune des lignes de sortie.

Si on veut comparer les différentes méthodes de hachage *sur des intervalles de temps plus longs*, il faut utiliser un logiciel de capture ne stockant pas les informations *paquet par paquet* mais les résumant *flux par flux*. Le risque principal est cependant une importante perte de précision. Dans le but de mesurer la perte de précision, nous avons, à partir de notre trace de départ, créé une trace stockant les informations *flux par flux*. Nos différents algorithmes de hachage ont alors été appliqués sur cette nouvelle trace. Les résultats obtenus montrent une répartition de trafic semblable à celle obtenue avec la trace stockant les informations *paquet par paquet*. Il paraît donc raisonnable d'envisager l'utilisation d'un logiciel stockant les informations *flux par flux* pour mesurer la répartition instantanée du trafic sur les différentes lignes de sortie durant des intervalles de temps relativement longs.

Table des matières

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Introduction

L'Internet est devenu en quelques années le premier réseau de communication de la planète et le moteur de la "Nouvelle Economie". Il connaît encore une croissance très importante et devrait à terme unifier toutes les communications humaines, voix, vidéo et données, et devenir un véritable réseau temps réel. Mais tout n'est pas si simple. Téléphonie et vidéoconférence, jeux en réseau, radio et télévision jouent des coudes pour se faire une place dans un réseau déjà bien encombré. La vente de musique en ligne, par téléchargement de fichiers MP3, suscite un intérêt grandissant et révolutionne les circuits de distribution traditionnels. La communauté des fournisseurs de service qui administrent l'INTERNET est confrontée à ce problème de croissance explosive. Des points de congestion peuvent notamment se développer dans le réseau. Une première manière d'éviter les points de congestion est d'augmenter la *capacité* du réseau. Celle-ci n'est cependant pas extensible à l'infini. Une autre manière de procéder est de *mieux répartir la charge de trafic* se trouvant à l'intérieur du réseau. Nous nous intéressons uniquement à cette dernière solution dans le cadre de cette étude. Nous nous intéressons plus particulièrement à la manière de répartir la charge de trafic dans les gros réseaux BACKBONE.

Les réseaux BACKBONE sont des réseaux à hauts débits constitués de lignes longue distance reliant ensemble d'autres réseaux de moins hauts débits. Nous nous intéressons aux réseaux BACKBONE "purement" IP, c'est-à-dire ceux effectuant du routage IP "classique". Le routage IP "classique" ne maintient pas d'états dans les routeurs. Quand un paquet IP arrive, le routeur regarde le contenu du paquet et choisit une ligne de sortie en tenant compte de sa *table de routage*. Celle-ci n'est rien d'autre qu'une base de données distribuées, mise à jour entre machines adjacentes. La mise à jour des tables de routage se fait de manière bien définie, grâce à un *protocole de routage*. Son objectif est de rechercher un ou plusieurs chemins de coût[1] minimum à travers le réseau. Les protocoles de routage les plus répandus *à l'intérieur* des réseaux BACKBONE sont les protocoles de routage ISIS[ORA90] et OSPF [MAL98] (cfr. chapitre 1). Nous examinons en particulier le protocole OSPF. Dans sa version OSPF*v1*, cet algorithme ne peut fournir qu'*un seul* chemin de coût minimum. Dans sa version OSPF*v2* [MOY91], cet algorithme *peut* fournir *plusieurs* chemins de coût minimum. Cette *possibilité* de trouver plusieurs chemins de coûts minimums est intéressante. Elle nous offre la *possibilité* de *répartir* la charge de trafic sur plusieurs chemins, c'est à dire d'effectuer du LOAD BALANCING. Le LOAD BALANCING devrait permettre d'éviter qu'un chemin puisse se trouver en situation de congestion alors que d'autres chemins restent inutilisés. Encore faut-il que la *probabilité* de trouver *plusieurs* chemins de coût minimum soit relativement importante et que la charge de trafic puisse être répartie *correctement* entre les différents chemins.

La probabilité de trouver plusieurs chemins de coût minimum dans un réseau BACKBONE INTERNET est heureusement relativement importante. Les réseaux BACKBONE comprennent effectivement le plus souvent de multiples connexions parallèles entre leurs points d'accès importants et ceci de manière à assurer la disponibilité à l'intérieur du réseau. La probabilité de trouver plusieurs chemins de coût minimum pourrait encore augmenter dans le futur. Les connexions parallèles se multiplient en effet au fur et à mesure du déploiement de la nouvelle technologie DWDM (DENSE WAVELENGHT DIVISION MULTIPLEXING) [BAL96] dans les réseaux

[1] Les coûts peuvent être les délais sur chacun des chemins, le nombre de sauts sur chacun des chemins, la charge des chemins, ...

BACKBONE. Le multiplexage en longueur d'onde permet de créer des dizaines voire même des centaines[2] de chemins entre les différents routeurs des réseaux BACKBONE.

Nous nous intéressons à la répartition de la charge de trafic *au niveau local*, c'est à dire sur les différentes *lignes* de sortie d'*un* routeur particulier du réseau BACKBONE. Le problème peut alors se résumer de la manière suivante : *Supposons n lignes sortant d'un routeur x. Supposons que parmi ces n lignes de sortie, il y en ait k qui soient sélectionnées par l'algorithme de routage et donc utilisables pour envoyer du trafic vers un point particulier y du réseau. Comment s'y prendre pour répartir équitablement le trafic entre les k lignes ? Quelle méthode de répartition de trafic utiliser ?*
Nous déterminons, au chapitre 2, les qualités essentielles d'une bonne méthode de répartition du trafic. Nous examinons ensuite, parmi les différentes méthodes de LOAD BALANCING existantes, celles qui, d'un point de vue théorique, sur du trafic *uniforme*[3], répondent le mieux à ces qualités. Nous verrons que certaines méthodes, dites méthodes de hachage s'avèrent être, en théorie, fort intéressantes. Si les méthodes de hachage répartissent bien le trafic *uniforme*, rien n'indique cependant qu'elles présentent les mêmes qualités en ce qui concerne le trafic *non uniforme*, comme celui que l'on risque de trouver dans les réseaux BACKBONE IP *réels*. C'est pourquoi une partie importante de cette étude consiste à tester les méthodes de hachage sur du trafic BACKBONE *réel*.

Nos algorithmes de hachage ne pouvant être implémentés directement dans un routeur d'un gros réseau BACKBONE, nous les *simulons* sur du trafic réel capté dans un gros réseau (cfr. chapitre 4). Nous les simulons tout d'abord sur une trace capturée avec un logiciel stockant les informations *paquet par paquet* et interprétons les résultats obtenus. Le stockage des informations *paquet par paquet* implique cependant un volume de données considérable et oblige dès lors à effectuer les mesures expérimentales sur des intervalles de temps relativement courts. Pour comparer les différentes méthodes de hachage *sur des intervalles de temps plus longs*, il vaut mieux utiliser un logiciel de capture de trafic ne stockant pas les informations *paquet par paquet* mais les *résumant* en mémoire. Une manière de procéder est de résumer les informations *flux par flux*[4]. Il risque cependant d'en résulter une importante perte de précision, comme expliqué au chapitre 3. Dans le but de mesurer la perte de précision, nous créons, à partir de la trace stockant les informations *paquet par paquet*, une nouvelle trace stockant les informations *flux par flux*. Nous appliquons ensuite nos algorithmes de hachage sur cette nouvelle trace et interpréterons les résultats obtenus.

L'élaboration de ce travail n'aurait sûrement pas été possible sans l'aide ni les conseils avisés de Monsieur Olivier BONAVENTURE, promoteur de ce mémoire. Je tiens à le remercier vivement. Ma gratitude va également à Steve UHLIG, mon voisin de bureau et chercheur au département Réseau de l'Institut d'Informatique, pour les conversations fructueuses que nous avons eues ensemble et qui m'ont permis de profiter pleinement de son expérience en matière de réseaux BACKBONE. Steve a également relu ce travail pour en relever les imperfections. Finalement, je ne peux oublier toute l'aide que ANNE, mon épouse, a pu m'apporter.

---

[2] ALCATEL a démontré à l'ensemble de la presse européenne qu'il était possible d'envoyer jusqu'à 240 canaux de 10 Gbps sur une même fibre optique [ALC00]

[3] Cette notion sera précisée au chapitre 4.

[4] Un *flux* correspond, dans cette étude, à un ou plusieurs paquets IP utilisant le même protocole IP, ayant les mêmes adresses IP source et destination et ayant les mêmes numéros de port TCP ou UDP.

Chapitre 1

Le LOAD BALANCING dans les réseaux BACKBONE

1.1 Introduction

Nous commençons par expliquer la notion de BACKBONE. Nous essayons ensuite de comprendre l'intérêt d'effectuer du LOAD BALANCING à l'intérieur de ce type de réseau. Nous nous intéressons ensuite aux protocoles de routage RIP, ISIS et OSPF utilisés *à l'intérieur* des réseaux INTERNET BACKBONE . Enfin, nous examinons dans quelle mesure les protocoles de routage RIP, ISIS et OSPF permettent d'effectuer du LOAD BALANCING.

1.2 Notion de BACKBONE

On peut simplifier l'INTERNET en le représentant comme un ensemble de réseaux hétéroclites connectés entre eux de manière plus ou moins *hiérarchique*. Au bas de l'échelle, on trouve les réseaux locaux (LANS) ; par exemple les ISP*s* locaux, les réseaux d'entreprises ou d'universités. Ces réseaux sont généralement connectés à l'INTERNET via des ISP*s*[5] régionaux (BELNET, EUNET, SKYNET, ...). Ces ISP*s* régionaux sont eux-mêmes connectés[6] à un ou plusieurs gros réseaux de type BACKBONE[7] (SPRINT, AT&τ, UUNET, vBNS, EBONE,...). Les réseaux BACKBONE sont essentiellement constitués de lignes longue distance à très hauts débits reliant ensemble d'autres réseaux de moins hauts débits. A titre d'exemple, la figure 1.1 [ATT00] montre le réseau BACKBONE de AT&T.

---

[5] Internet Service Provider. Société donnant accès à Internet moyennant un abonnement payant ou gratuit

[6] La séparation entre ISPs régionaux et réseaux de type "BACKBONE" n'est pas toujours si évidente. De plus en plus de réseaux régionaux sont effectivement connectés entre eux via des NAPs (Network Access Points). Le trafic peut alors passer à travers une série de réseaux régionaux et ne plus passer par les réseaux BACKBONE.

[7] Colonne vertébrale, épine dorsale.

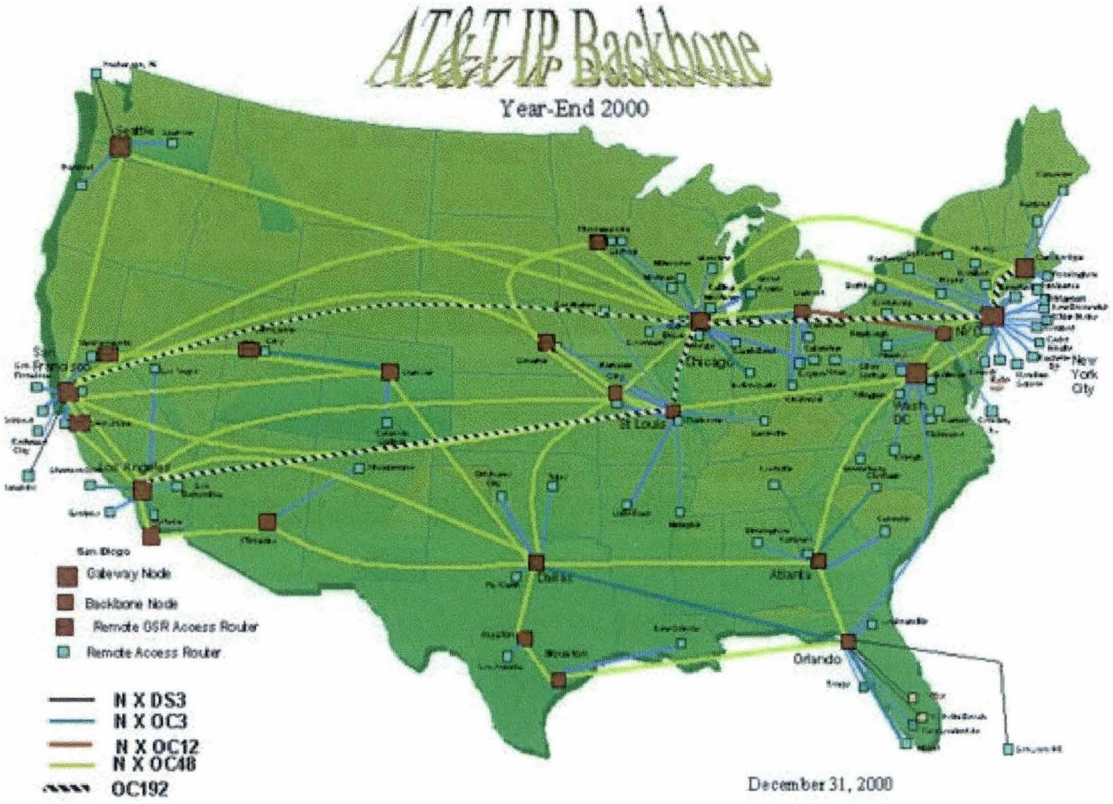


Figure 1.1 :
Réseau BACKBONE de AT&T.

OC signifie "OPTICAL CARRIER " (ligne de transmission sur fibre optique). Une ligne OC1 offre un débit de 51,84 Mbits/s. Une ligne OC3 un débit triple, …, une ligne OC48 un débit de 2,5 Gbps et une ligne OC192 un débit de 10 Gbps.

1.3 Intérêt du LOAD BALANCING

Le LOAD BALANCING devrait permettre d'éviter qu'un chemin d'un réseau BACKBONE puisse se trouver en situation de congestion alors que d'autres chemins restent inutilisés. Prenons l'exemple de figure 1.2 en supposant que toutes les lignes ont un poids égal à 1.

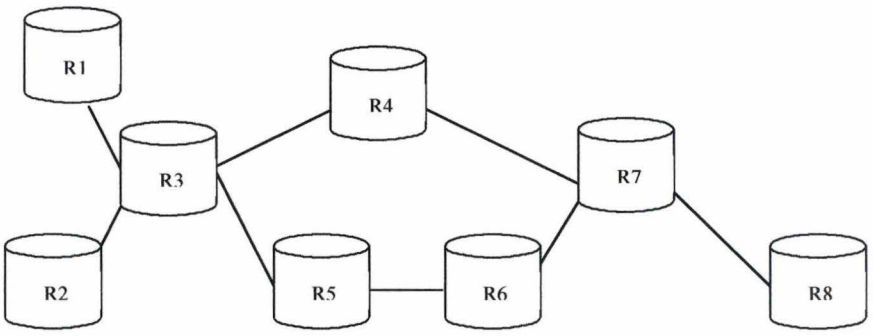


Figure 1.2: Problème du "poisson"

Supposons que le plus court chemin pour aller de R1 à R8 passe par les routeurs R3, R4 et R7. Le plus court chemin entre R2 et R8 passe dès lors par les mêmes routeurs. Le trafic (dans le sens gauche-droite du dessin) passe donc toujours par le même chemin. En cas de forte charge du réseau, les lignes R3-R4 et R4-R7 pourraient rapidement devenir saturées. Il se produirait donc une congestion dans le réseau alors qu'il existe pourtant une autre ligne non saturée permettant de relier les routeurs R1 et R2 au routeur R8. Il serait donc intéressant d'acheminer une partie du trafic également sur les lignes R3-R5, R5-R6 et R6-R7.

1.4 Les protocoles de routage *à l'intérieur* des réseaux BACKBONE

Dans l'INTERNET, les machines communiquent entre elles en s'envoyant des messages. Ces messages contiennent entre autres l'adresse de l'émetteur et l'adresse du destinataire. Sur base de ces informations, les paquets[8] sont acheminés à travers tout le réseau. Deux méthodes peuvent être utilisées pour véhiculer l'information:

1) *par circuit virtuel.* On crée ici un *circuit* entre l'émetteur et le destinataire et tous les paquets entre les deux stations ne passent que par ce circuit. Ceci nécessite de maintenir des *états* dans les routeurs. Un *état* comprend, pour chaque flux, la réservation des *ressources nécessaires, ainsi qu'une ligne de sortie unique, définie une fois pour toutes.*

[8] Le paquet est l'unité de transmission au niveau du protocole IP.

2) *par* paquet. Ici, aucun circuit n'est créé. Aucun état n'est maintenu dans les routeurs. Le choix de la ligne de sortie est effectué à chaque arrivée d'un paquet IP. Il est effectué en fonction du contenu du paquet et du contenu de la table de routage. Deux paquets concernant une même liaison entre deux machines peuvent donc cheminer de manière différente dans l'INTERNET en fonction des décisions prises par les routeurs. La plupart des protocoles de routage dans l'INTERNET sont basés sur cette approche.

Nous nous intéressons au routage *par paquet*. Au niveau de ce type de routage, deux options peuvent être choisies : le routage statique et le routage dynamique.

1.4.1 Routage statique

La manière dont les paquets doivent être routés est définie manuellement au niveau de chaque routeur dans une table de routage statique. Le routage statique ne permet pas de supporter plusieurs routes actives. Il ne permet donc pas le LOAD BALANCING. Il empêche également toute réaction rapide et automatique en fonction des changements de la topologie du réseau. Cette méthode est donc rarement utilisée, sauf dans certains réseaux IP très stables.

1.4.2 Routage dynamique

Ici, les routeurs apprennent "rapidement" et "automatiquement" toutes les modifications topologiques. Pour ce faire, ils collaborent entre eux de manière à ce que chacun obtienne finalement une image "correcte" du réseau. Les tables de routage de ces routeurs sont donc dynamiques. Elles constituent une base de données distribuées, mise à jour entre machines voisines. La *mise à jour* des tables de routage se fait de manière bien définie, grâce à un ou plusieurs *protocoles de routage*. Les protocoles de routage IP utilisent des *algorithmes de routage* dont l'objectif est de rechercher un ou plusieurs chemins de *coût minimum* à travers le réseau.
Deux types de protocoles de routage sont utilisés pour connaître la topologie d'un réseau et calculer les meilleurs chemins permettant d'atteindre une destination. Il s'agit des protocoles de routage à *vecteur de distance* et des protocoles de routage à *état de liaison*.

1.4.2.1 Les protocoles de routage à vecteur de distance

L'algorithme de routage à vecteur de distance est basé sur le fait que chaque routeur dispose d'une table de routage précisant pour chaque destination le meilleur coût connu et par quel chemin l'atteindre (c'est le vecteur de distance). Chaque routeur envoie périodiquement à ses voisins des *messages de signalisation* comprenant la liste de ses vecteurs de distance. Chaque routeur recevant une telle liste peut donc mettre à jour sa table de routage. Pour chaque destination pour laquelle le coût local est supérieur à celui reçu, l'entrée correspondante vers cette destination dans la table de routage locale est modifiée. Cette méthode permet ainsi de converger vers un ensemble de chemins optimaux.

RIPv2 (ROUTING INFORMATION PROTOCOL) [MAL98] est un protocole de routage à vecteur de distance utilisé *à l'intérieur* des réseaux BACKBONE. Les coûts sont calculés en fonction du nombre de sauts, c-à-d en nombre de lignes intermédiaires entre la source et la destination.

L'algorithme de routage utilisé est l'algorithme BELLMAN-FORD [MAL 98]. Pour des raisons de performance, le nombre maximum de sauts a été fixé à 15. Ceci ne le rend pas très intéressant pour de gros réseaux. RIP est en fait idéal pour de réseaux de taille *moyenne*. Son principal avantage réside dans sa *facilité* d'implémentation. Ses inconvénients sont essentiellement les suivants :

1) il ne peut s'adapter à des changements dans la charge du réseau
2) il ne peut en principe pas supporter plusieurs routes actives (des exceptions existent [THA00]). Il ne permet donc en principe pas le LOAD BALANCING.
3) la route comptant le moins de sauts est toujours choisie même si un routage plus efficace est possible
4) il peut relativement vite saturer le réseau par les nombreux paquets qu'il envoie pour mettre à jour les tables de routage.

1.4.2.2 Les protocoles de routage à état de liaison

Les protocoles à *état de liaison* permettent à chaque routeur de posséder une image complète de la topologie du réseau. Avec ce protocole, un routeur n'échange pas des distances avec ses voisins. Au lieu de cela, chaque routeur teste l'état de la liaison entre chacun de ses voisins, envoie cette information via des messages d'état de liaison (LSA - LINK STATE ADVERTISEMENT) à ses autres voisins, qui ensuite la propagent dans tout le réseau. Chaque routeur capture cette information d'état de liaison et construit une table de routage complète. Chaque routeur possède donc une image complète de la topologie du réseau grâce à l'envoi régulier par les autres routeurs de leur structure locale. Ensemble, tous les états de liaison forment la base de données d'état de liaison qui décrit le réseau et permet ainsi de calculer les chemins. L'algorithme SPF ou "le chemin le plus court d'abord" calcule le chemin le plus court entre un routeur "source" et tous les autres routeurs du réseau. Les deux principaux protocoles de routage à état de liaison utilisés *à l'intérieur* des réseaux BACKBONE sont les protocoles ISIS[9] [ORA90] et OSPF[10] [MOY89]. Dans ces deux protocoles, c'est l'algorithme de DIJKSTRA qui est utilisé. Cet algorithme très efficace[11] peut s'exécuter rapidement dans les ordinateurs actuels. De plus, tous les routeurs disposant de la même base de données, il ne peut y avoir de boucle : une fois l'inondation et les calculs terminés, toutes les routes sont "saines".

D'un point de vue pratique, il est important de savoir qu'un protocole d'état de liaison converge plus vite qu'un protocole à vecteur de distance [PER92]. Par *converger*, nous voulons dire se stabiliser après que quelque chose ait changé, comme un routeur ou une liaison qui tombe en panne. OSPF et ISIS comprennent d'autres avantages par rapport à RIP[PER92].

---

[9] INTERMEDIATE SYSTEM TO INTERMEDIATE SYSTEM
[10] OPEN SHORTEST PATH FIRST
[11] L'algorithme de DYKSTRA présente une complexité de l'ordre de O (m*log m) où m représente le nombre de lignes dans le réseau.

1.5 Le LOAD BALANCING dans les réseaux BACKBONE

La charge du trafic peut être distribuée à différents niveaux. Nous parlons brièvement ci-dessous des niveaux 2 (couche liaison de données) et 3 (couche réseau).

1.5.1 Obtention de plusieurs chemins de coût minimum au niveau de la couche 2

Une manière intelligente de distribuer le trafic est d'utiliser les circuits virtuels de la couche 2. Ici, les routeurs IP sont placés en périphérie d'un réseau BACKBONE dont le centre est constitué de switch ATM[12] ou FRAME RELAY[13] (figure 1.3).

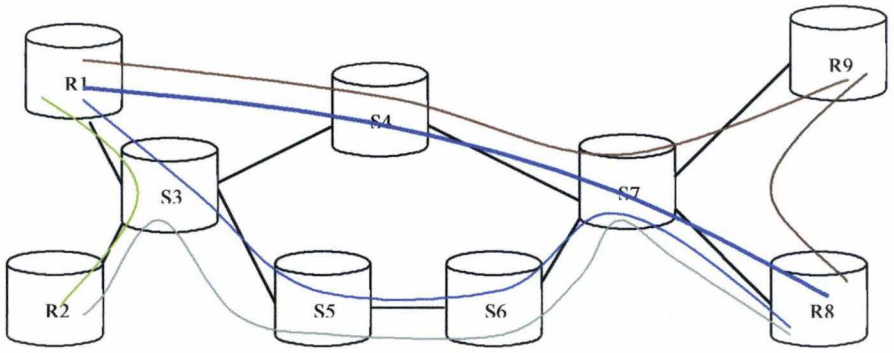


Figure 1.3: Circuits virtuels
R1, R2,R8 et R9 sont les routeurs IP
S3, S4, S6, S6 et S7 sont les switches ATM ou FRAME RELAY

Des connexions virtuelles existent entre chaque paire de routeurs IP. Ainsi, R1 pense qu'il est directement connecté avec R8 et R9. Deux circuits redondants (les circuits en bleu) permettent d'acheminer du trafic de R1 vers R8.

Cette solution, très séduisante, fonctionne dans beaucoup de réseaux. Elle pose cependant des problèmes parce qu'elle oblige à gérer deux réseaux à la fois : le réseau IP et le réseau ATM (ou FRAME RELAY) qui se trouve au cœur du BACKBONE. Cette solution risque donc de multiplier par deux les problèmes potentiels. Ceci, les opérateurs de réseau veulent l'éviter. Vu l'évolution du protocole IP, ils peuvent à présent se permettre de placer des routeurs IP (niveau 3) au cœur de leur réseau, ce qui leur permet d'obtenir un réseau complètement IP. *Le sujet de notre étude portant uniquement sur les réseaux* BACKBONE INTERNET *"purement" IP, nous ne parlerons plus de l'acheminement au niveau 2 dans la suite de cette étude.*

[12] ASYNCHRONOUS TRANSFER MODE. Mode de transmission fonctionnant par commutation de paquets fixes, appelés *cellules*.

[13] Relais de trame. Sorte de commutation de paquets simplifiée (sans contrôle de flux ni d'erreur) de façon à assurer une commutation plus rapide.

1.5.2 Obtention de plusieurs chemins de coût minimum au niveau de la couche 3

Nous avons vu à la section 1.4.2.1 que le protocole RIP ne supporte pas, en principe, plusieurs routes actives. Il ne permet donc en principe pas le LOAD BALANCING.

Les protocoles de routage à état de liaison sont eux, par contre, capables d'effectuer du LOAD BALANCING. Nous nous intéressons plus particulièrement au protocole OSPF dans le cadre de cette étude. Ce protocole est disponible en deux versions. La première version, OSPF v1 [MOY 89], sélectionne aléatoirement *un* des chemins trouvés. La deuxième version, OSPF v2, est celle qui nous intéresse car elle permet, grâce à ECM (EQUAL COST MULTIPATH) [HOP00], de répartir *équitablement* le trafic sur plusieurs lignes de sortie du routeur. Cette répartition s'effectue cependant *sans concertation* avec les autres routeurs. Le trafic n'est donc réparti équitablement que *localement*. Ceci risque de créer des problèmes. A titre d'exemple, reprenons la figure 1.2 et supposons que les lignes R3-R4 et R3-R5 soient de charges égales et ne soient pas saturées. Supposons également que le routeur R6 soit en congestion. Avec ECM, le routeur R3 va répartir le trafic de manière égale entre les lignes R3-R4 et R3-R5. La plus grosse partie du trafic en provenance de la ligne R3-R5 risque donc d'être perdue parce que détruite par le routeur R6.
Pour éviter ceci, il faudrait faire en sorte que les routeurs soient informés de la charge présente sur les différents chemins du réseau. Ceci est possible avec l'extension OMP (OPTIMIZED MULTIPATH) [VIL99] de OSPF. Cette extension permet d'informer chaque routeur de la charge actuelle dans tout le réseau et de calculer de manière autonome la répartition des paquets sur plusieurs chemins afin d'optimiser la charge complète du réseau tout en évitant les oscillations.

1.5.3 Probabilité d'obtention de plusieurs chemins de coût minimum

La probabilité de trouver plusieurs chemins de coût minimum est heureusement relativement importante. Ceci est dû d'une part, à la volonté des gestionnaires de réseau de créer des chemins redondants dans leur réseau de manière à offrir ainsi une plus grande sécurité d'utilisation et, d'autre part, à l'accroissement constant du trafic qui nécessite l'ajout régulier de nouveaux chemins (généralement, quand une ligne est occupée à plus de 50%, le gestionnaire de réseau crée une nouvelle ligne redondante). Les chemins redondants dans le réseau peuvent être créés en jouant, par exemple, sur les coûts associés aux différentes lignes. Ainsi, en ce qui concerne le réseau de la figure 1.2, le gestionnaire pourrait créer une redondance en attribuant un poids égal à 0 à la liaison R5-R6. Il pourrait également attribuer un poids égal à 2 à la liaison R4-R7. Une manière de créer des lignes redondantes est donc de jouer intelligemment avec les coûts associés aux différentes lignes.

1.6 Conclusions

Nous avons montré l'intérêt de répartir la charge de trafic sur différentes lignes dans les réseaux BACKBONE. Ceci permet essentiellement d'éviter qu'un chemin puisse se trouver en situation de congestion alors que d'autres chemins restent inutilisés. Nous avons vu que l'extension OMP du protocole OSPFv2 permettait de trouver plusieurs chemins de coût minimum. *Encore faut-il répartir correctement la charge de trafic entre les différents*

chemins. Nous verrons au chapitre 2 les qualités essentielles que doit posséder toute bonne méthode de répartition de trafic.

Chapitre 2

Les méthodes de répartition du trafic

2.1 Introduction

Les méthodes de répartition du trafic ont pour objectif essentiel d'éviter les points de congestion dans le réseau en répartissant le plus équitablement possible, sur plusieurs chemins, le trafic devant être acheminé d'un routeur *x* vers un point particulier *y* se trouvant à une certaine distance dans le réseau. Dans cette étude, nous nous intéressons à la répartition du trafic au niveau *local*, c'est-à-dire sur les différentes *lignes* de sortie d'*un* routeur particulier d'un réseau BACKBONE. Le problème peut se résumer de la manière suivante : *supposons n lignes sortant du routeur x. Supposons que parmi ces n lignes de sortie, il y en ait k qui soient sélectionnées par l'algorithme de routage et donc utilisables pour envoyer du trafic vers le point particulier y du réseau. Supposons également que nous connaissions la clé de répartition du trafic entre les différentes lignes sélectionnées. Par exemple, 25% du trafic devrait passer par la première ligne sélectionnée, 30% du trafic par la deuxième , etc. D'autres clés de répartition sont bien entendu possibles. Comment s'y prendre pour distribuer le trafic entre les k lignes ? Quelle méthode de répartition utiliser ?*

Pour répondre à la question, nous commençons par déterminer, à la section 2.2, les qualités essentielles d'une bonne méthode de répartition de trafic. Nous examinons ensuite (sections 2.3 et 2.4), parmi les différentes méthodes de répartition de trafic existantes, celles qui, d'un point de vue théorique, sur du trafic uniforme[14], répondent le mieux à ces qualités.

2.2 Qualités essentielles d'une bonne méthode de répartition de trafic

Une bonne méthode de répartition de trafic doit répartir la charge de trafic le plus équitablement possible entre les différentes lignes de sortie choisies par l'algorithme de routage. Elle doit également respecter l'ordre dans lequel les paquets arrivent à destination et doit pouvoir être implémentée dans les routeurs des réseaux BACKBONE fonctionnant à hauts débits.

[14] Il s'agit ici de l'uniformité des valeurs de la clé utilisée pour répartir le trafic (cfr. section 2.3.2.1).

2.2.1 Une bonne méthode de répartition de trafic doit répartir le plus équitablement possible la charge de trafic entre les différentes lignes de sortie choisies par l'algorithme de routage

Il faut effectivement éviter au maximum les points de congestion dans le réseau. Un routeur entre en congestion lorsqu'il reçoit une charge de trafic trop importante par rapport à sa capacité de traitement ou de mémorisation. Des pertes de paquets se produisent alors dans le routeur. Le nombre de points de congestion dans un réseau BACKBONE risque d'être d'autant plus important que le trafic est mal réparti à l'intérieur du réseau.

Des mesures concernant le type de trafic présent dans les réseaux BACKBONE IP ont été effectuées [THOM 98]. Ces mesures ont montré que le protocole TCP [POS81-b] était de loin le protocole le plus répandu dans ce type de BACKBONE. Sur une période de mesure d'une journée, 95% des bytes, 90% des paquets et 80% des flux ont correspondu, en moyenne, à du trafic TCP. Ces chiffres ne sont bien entendu pas à prendre "à la lettre" puisqu'ils ne concernent qu'*une* mesure dans *un* réseau BACKBONE durant *une* période de temps déterminée. Il n'empêche qu'ils sont assez représentatifs de l'importance du protocole TCP. L'importance de ce protocole a été mise en évidence dans d'autres études [CLA 00].

Un inconvénient majeur du protocole TCP est qu'il peut provoquer un écroulement de la performance globale d'un réseau chargé si celui-ci comprend un nombre trop important de points de congestion. Ceci résulte de l'algorithme CONGESTION AVOIDANCE de TCP. L'idée de cet algorithme est qu'une indication de pertes de paquets indique une situation de congestion dans un ou plusieurs routeurs. Une perte de paquets peut être signalée à tout moment par un TIMEOUT ou par la réception d'un certain nombre d'acquits dupliqués par la destination. En cas de présence de points de congestion, chaque flux TCP actif va détecter des pertes et agir en conséquence. Ainsi, les centaines ou milliers de flux TCP simultanés vont repasser en mode SLOW START et tenter de réémettre les paquets perdus, contribuant à accroître la congestion. Ce phénomène est connu sous le nom de GLOBAL SYNCHRONIZATION et met en évidence l'importance de techniques de prévention de la congestion.

Une bonne méthode de LOAD BALANCING doit donc *répartir,* de la manière la plus *équitable* possible, la charge de trafic entre les différentes lignes, de manière à éviter les points de congestion. Ceci nécessite un découpage du trafic en morceaux les plus petits possibles, c'est-à-dire un découpage en paquets IP. Lors de chaque arrivée d'un paquet IP, la méthode de répartition doit calculer une ligne de sortie sur laquelle le paquet IP va être envoyé. La manière dont les lignes de sortie sont calculées doit permettre de garantir une bonne répartition du trafic tout en tenant compte des débits des différentes lignes de sortie.

2.2.2 Une bonne méthode de répartition de trafic doit respecter l'ordre dans lequel les paquets arrivent à destination

Un réordonnancement des paquets fait effectivement croire à TCP que des paquets ont été perdus dans le réseau, ce qui fausse les mécanismes de contrôle de congestion. TCP va dès lors retransmettre inutilement des paquets, ce qui va augmenter le trafic et donc provoquer une dégradation inutile des débits réels [JAC90] [BRA94]. Ceci est évidemment exactement le contraire de ce que l'on veut obtenir en effectuant du LOAD BALANCING.

2.2.3 Une bonne méthode de répartition de trafic doit pouvoir être implémentée dans les routeurs des réseaux BACKBONE fonctionnant à hauts débits.

Elle doit donc reposer sur un *algorithme simple,* n'effectuant pas un trop grand nombre de calculs pour déterminer sur quelle ligne de sortie un paquet IP doit être envoyé. L'idéal serait bien entendu d'utiliser un algorithme simple déjà implémenté dans la plupart des routeurs.

2.3 Les différentes méthodes existantes de répartition du trafic

Il existe deux grands groupes de méthodes de répartition de trafic :

1) les méthodes distribuant les paquets IP *indépendamment de leur contenu.* Ces méthodes tiennent compte de facteurs macroscopiques concernant les paquets IP tels que leur ordre d'arrivée, leur taille, ... Elles peuvent éventuellement tenir compte de la *clé de répartition*[15] de la charge de trafic entre les différentes lignes de sortie sélectionnées par l'algorithme de routage. Elles n'offrent cependant aucune garantie concernant l'*ordre* dans lequel les paquets arrivent à destination.
2) les méthodes distribuant les paquets IP *en fonction de leur contenu.* Ces méthodes ne prennent pas en considération tout le contenu du paquet IP mais uniquement un ou plusieurs champs caractérisant le *flux*[16] auquel appartient le paquet IP. Elles offrent des garanties concernant l'*ordre* dans lequel les paquets arrivent à destination. Elles peuvent également, moyennant quelques petites modifications, tenir compte de la clé de répartition du trafic entre les différentes lignes de sortie sélectionnées .

2.3.1 Méthodes distribuant les paquets indépendamment de leur contenu

Ces méthodes sont basées sur l'algorithme ROUND ROBIN.

a. *L'algorithme* ROUND ROBIN permet de distribuer les paquets IP de manière *simple* à travers plusieurs lignes de sortie. Il repose sur le principe du "tourniquet". Les différentes lignes de sortie du routeur sont visitées à tour de rôle par le "tourniquet". Chaque paquet arrivant dans le routeur est expédié sur la ligne de sortie correspondant à la position du tourniquet. Le tourniquet est ensuite incrémenté d'une position. Cette façon de répartir le trafic n'est évidemment *pas très équitable* si les différents flux comportent des paquets de tailles variables. Or, les réseaux BACKBONE réels, sujets de notre étude, comportent précisément des paquets de tailles variables [MIL98]. C'est la raison pour laquelle il est préférable d'utiliser un algorithme basé sur la méthode DEFICIT ROUND ROBIN [SHR94].

b. L'algorithme DEFICIT ROUND ROBIN permet de tenir compte des paquets de tailles variables. Cet algorithme repose également sur le principe du "tourniquet". Il associe à chaque ligne de sortie *i* un compteur *d(i)* supérieur ou égal à zéro, appelé "*déficit* de la ligne de sortie". A chaque passage du "tourniquet" devant une ligne de sortie, le déficit d(i) de cette ligne est incrémenté d'une quantité constante appelée *quantum.* Le paquet en tête de la ligne d'entrée

[15] La *clé de répartition* définit le pourcentage de trafic devant être octroyé aux différentes lignes sélectionnées

[16] Un *flux* correspond, dans cette étude, à un ou plusieurs paquets IP utilisant le même protocole IP, les mêmes adresses source et destination et les mêmes numéros de ports TCP ou UDP.

n'est envoyé sur la ligne de sortie sélectionnée par le tourniquet que si sa taille est inférieure ou égale au déficit de cette ligne de sortie.

Un exemple concret expliquant la manière dont fonctionne DEFICIT ROUND ROBIN est expliqué aux figures 2.1, 2.2, 2.3, 2.4 et 2.5 suivantes. Supposons un routeur comprenant quatre lignes de sortie au démarrage de l'algorithme DEFICIT ROUND ROBIN. Supposons un certain nombre de paquets présents sur la ligne d'entrée. Le paquet en tête de la ligne d'entrée a une taille de 160 bytes. Les paquets qui suivent sur la ligne d'entrée ont des tailles respectives de 400, 300, 550 et 950 bytes (figure 2.1). Au départ, les déficits d(i) de toutes les lignes de sortie sont égaux à 0.

Ligne d'entrée	...	...	950	550	300	400	160

Ligne de sortie 1					d(1)=0
Ligne de sortie 2					d(2)=0
Ligne de sortie 3					d(3)=0
Ligne de sortie 4					d(4)=0

Figure 2.1

Quand le tourniquet arrive à la ligne de sortie n°1 (figure 2.2), le déficit *d(1)* de cette ligne est incrémenté d'une quantité égale au *quantum* (qui vaut 500 dans cet exemple). Le déficit *d(1)* vaut donc à présent 500.

Ligne d'entrée	...	...	950	550	300	400	160

→

ligne de sortie 1							d(1)=500
Ligne de sortie 2							d(2)=0
Ligne de sortie 3							d(3)=0
Ligne de sortie 4							d(4)=0

Figure 2.2

La taille du paquet en tête de la ligne d'entrée étant ≤ 500, la ligne de sortie n°1 est autorisée à le recevoir. Elle ne peut recevoir un deuxième paquet de la ligne d'entrée, la taille totale des deux premiers paquets dépassant 500 bytes. Après avoir expédié le paquet sur la ligne de sortie, le routeur décrémente *d(1)* de 160 bytes (figure 2.3).

Ligne d'entrée	...	...	...	950	550	300	400

→

Ligne de sortie 1	160						d(1)=340
Ligne de sortie 2							d(2)=0
Ligne de sortie 3							d(3)=0
Ligne de sortie 4							d(4)=0

Figure 2.3

Le tourniquet arrive ensuite à la *ligne n°2* et ainsi de suite jusqu'à la ligne de sortie n°4. La ligne de sortie n°4 ne reçoit pas de paquet, son déficit étant inférieur à 550 (figure 2.4).

Ligne d'entrée	...	...	...	...	...	950	550

	Ligne de sortie 1				160			d(1)=340
	Ligne de sortie 2			400				d(2)=100
	Ligne de sortie 3		300					d(3)=200
→	Ligne de sortie 4	-						d(4)=500

Figure 2.4

Au deuxième passage du tourniquet sur la *ligne de sortie n°1*, d(1) est égal à 840, ce qui permet au routeur d'envoyer le paquet de 550 bytes sur cette ligne de sortie (figure 2.5). Le paquet suivant, de 950 bytes, ne peut être reçu sur les lignes de sortie 2 et 3 mais est reçu sur la ligne 4.

Ligne d'entrée	...	...	...	...	...	...	950

	Ligne de sortie 1				550				160	d(1)=290
	Ligne de sortie 2			-				400		d(2)=600
	Ligne de sortie 3		-				300			d(3)=700
→	Ligne de sortie 4	950				-				d(4)=50

Figure 2.5

La différence essentielle avec l'algorithme ROUND ROBIN réside donc dans le fait que *si une ligne de sortie n'est pas capable de recevoir un paquet lors du passage du tourniquet, elle augmente ses chances de recevoir un ou plusieurs paquets lors des passages suivants* et ceci grâce à l'augmentation de son déficit *d(i)*. Ainsi, les lignes de sortie ayant été peu chargées durant un tour obtiennent des compensations lors des tours suivants. Le déficit *d(i)* d'une ligne de sortie correspond en quelque sorte à "l'iniquité" produite dans le passé.

L'algorithme DEFICIT ROUND ROBIN améliore donc l'algorithme ROUND ROBIN pour des paquets de tailles variables. Il tend à un *partage* plus *équitable des paquets* IP *entre les différentes lignes de sortie*. Nous le généralisons dans le paragraphe suivant, de manière à tenir compte de la *clé de répartition* du trafic entre les différentes lignes de sortie sélectionnées par l'algorithme de routage.

c. *L'algorithme WEIGHTED ROUND ROBIN* permet d'attribuer à chaque ligne de sortie un *poids* en fonction de son débit maximal. Les différentes lignes de sortie sont placées sur le tourniquet de ROBIN en fonction du poids qui leur est attribué. La figure 2.6 représente le tourniquet dans le cas de quatre lignes de sortie *L1, L2, L3* et *L4*. Les débits maximums des quatre lignes de sortie valent respectivement *2*, *2*, *4* et *8* Mbits.

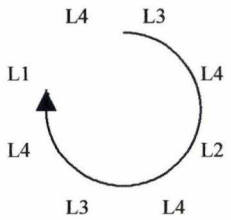


Figure 2.6: Le tourniquet de WEIGHTED ROUND ROBIN

WEIGHTED ROUND ROBIN répond à la plupart des qualités essentielles d'une bonne méthode de répartition du trafic:

1) Il permet une répartition relativement équitable des paquets IP en fonction des débits des différentes lignes de sortie. On peut même imaginer de l'implémenter *dynamiquement* de manière à tenir compte des pertes de paquets éventuellement constatées sur les différentes lignes de sortie. Si une perte importante de paquets IP était par exemple constatée sur la ligne de sortie *i*, ce qui est une indication de congestion, il y aurait réajustement automatique des poids (et donc du tourniquet) en défaveur de cette ligne *i*, limitant ainsi le nombre de paquets envoyés sur cette ligne *i*. Les poids attribués aux différentes lignes de sortie ne seraient donc pas établis une fois pour toutes mais seraient ajustés en fonction du nombre de paquets IP perdus sur les différentes lignes.
2) Il s'agit d'un algorithme simple, pouvant être implémenté dans les routeurs à hauts débits.

WEIGHTED ROUND ROBIN est cependant à proscrire en ce qui concerne le LOAD BALANCING. Cet algorithme n'offre effectivement *aucune garantie concernant l'ordre* dans lequel les paquets arrivent à destination. Les paquets issus d'un même flux risquent en effet d'emprunter des lignes différentes, ce qui risque d'entraîner des délais de transmission différents[17]. Des délais de transmission différents peuvent entraîner des réarrangements de paquet. Ainsi, l'ordre dans lequel les paquets arrivent à destination risque de ne plus correspondre à celui dans lequel les paquets quittent la source. Ceci fausse les mécanismes de contrôle de congestion, ce qui oblige TCP à retransmettre inutilement des paquets, provoque une augmentation du trafic et donc une *dégradation inutile des débits réels*.

2.3.2 Méthodes distribuant les paquets IP en tenant compte de leur en-tête

Il est impératif de diriger les paquets *issus d'un même* flux vers la *même* ligne de sortie du routeur. Il faut donc utiliser des méthodes permettant aux routeurs de connaître en tout ou en partie le flux auquel appartient un paquet IP qui arrive. Suivant le flux auquel il appartient, un paquet IP va être dirigé vers une ligne de sortie bien précise. Deux types de méthodes existent essentiellement dans cette catégorie :

2.3.2.1 Les méthodes déterministes (méthodes de hachage)

a. Principe général

Les méthodes de hachage déterminent la ligne de sortie d'un paquet IP en effectuant un calcul basé *uniquement* sur le contenu de l'en-tête de ce paquet IP. Pour ce faire, elles utilisent une fonction *h*, dite *fonction de hachage*, qui, à partir du contenu d'un paquet entrant dans le routeur, fournit un nombre entier $i = h(c)$ compris entre 1 et n, où n est le nombre de lignes de sortie sélectionnées et c est la *clé* de hachage. La clé de hachage est constituée de bits issus de l'en-tête du paquet IP. Elle n'est pas constituée de *tous* les bits de l'en-tête du paquet IP mais seulement d'une partie.

[17]La charge des différentes lignes de sortie peut effectivement varier au cours du temps et être différente d'une ligne de sortie à l'autre. Les délais d'attente des paquets dans les routeurs peuvent être différents.

b. Qualités essentielles d'une bonne méthode de hachage

Une bonne méthode de hachage doit satisfaire aux qualités essentielles d'une bonne méthode de répartition de trafic. Elle doit donc éviter de réordonner les paquets, doit répartir la charge de trafic le plus équitablement possible et pouvoir être implémentée dans les routeurs BACKBONE fonctionnant à hauts débits.

1. Non-réordonnancement des paquets

Il n'y aura pas de réordonnancement de paquets si la fonction de hachage est *déterministe* et utilise une *clé de hachage ne comprenant que des bits extraits de la représentation binaire du flux*. Dans ce cas, *tous* les paquets *issus d'un même* flux se dirigent effectivement vers la *même* ligne de sortie du routeur. Il n'y a donc pas réordonnancement de paquets.

De nombreuses *clés de hachage* sont évidemment possibles. Elles peuvent être constituées :

- des bits de l'adresse IP source uniquement,
- des bits de l'adresse IP destination uniquement,
- des bits du port source uniquement,
- des bits du port de destination uniquement,
- des bits de l'adresse IP source *et* des bits de l'adresse IP destination,
- ...

2. Répartition uniforme des paquets

Une bonne méthode de hachage doit répartir le trafic le plus uniformément possible. Les méthodes de hachage utilisant une *fonction de hachage " uniformément distribuée"*, c'est-à-dire une fonction de hachage qui tend à répartir les paquets de la manière la plus uniforme possible, semblent donc intéressantes. Il y a également intérêt à ce que le *domaine*[18] de la fonction de hachage soit le plus grand possible et que les différentes valeurs possibles de la *clé de hachage* soient réparties le plus uniformément possible.

Notre étude portant sur les réseaux BACKBONE réels, nous éliminons d'office les méthodes de hachage prenant comme clé de hachage uniquement le *port* source ou le *port* destination. L'ensemble des valeurs possibles pour la clé est effectivement fort restreint [CLA00][MIL98]. Les différentes valeurs possibles pour la clé sont de plus loin d'être uniformément réparties (par exemple, à peu près 85% du trafic est du trafic TCP) [CLA00].

[18] l'ensemble des valeurs possibles de la clé de hachage

3. Traitement des paquets à hauts débits

Une bonne méthode de hachage doit reposer sur un algorithme simple, n'effectuant pas un trop grand nombre de calculs pour déterminer sur quelle ligne de sortie un paquet IP doit être envoyé. L'idéal serait bien entendu d'utiliser un algorithme simple déjà implémenté dans la plupart des routeurs.

c. Les meilleures méthodes de hachage pour un trafic uniformément distribué

Un certain nombre d'études ont été effectuées sous la supposition que les *données* utilisées par les fonctions de hachage étaient *uniformément distribuées* [FLE83][GRE92]. Il ressort de ces études que les meilleures méthodes de hachage *pour un trafic uniformément distribué*, sont les méthodes de hachage suivantes :

- IP DESTINATION: méthode de hachage utilisant comme clé les 32 bits de l'adresse IP de destination (cfr. section 2.4.1).
- XOR IP DESTINATION : méthode de hachage utilisant comme clé les 8 bits du *"ou exclusif"* sur l'adresse IP de destination (cfr. section 2.4.2).
- XOR IP SOURCE DESTINATION : méthode de hachage utilisant comme clé les 8 bits du *"ou exclusif"* sur l'adresse IP source et l'adresse IP de destination (cfr. section 2.4.3).
- INTERNET CHECKSUM : méthode de hachage utilisant comme clé les 104 bits de la représentation binaire du flux[19] et utilisant comme fonction de hachage l'algorithme calculant la somme de contrôle des paquets IP, TCP et UDP (cfr. section 2.4.4).
- CRC: méthode de hachage utilisant comme clé les 104 bits de la représentation binaire du flux et utilisant comme fonction de hachage l'algorithme CRC. L'algorithme CRC calcule la somme de contrôle permettant de détecter les erreurs dans les trames (cfr. section 2.4.5).

d. Conclusions

Les méthodes de hachage décrites ci-dessus respectent bien l'ordre dans lequel les paquets arrivent à destination. Leur clé de hachage ne comprend en effet *que* des bits extraits de la représentation binaire du flux. Tous les paquets issus d'un même flux se dirigent donc vers la même ligne de sortie.
Elles répartiraient la charge de trafic de manière quasi parfaite sur les différentes lignes de sortie si les différentes valeurs possibles pour la clé de hachage étaient uniformément réparties. Dans les réseaux BACKBONE IP, *le domaine* de la fonction de hachage *et l'uniformité des valeurs possibles de la clé de hachage risquent cependant de fortement varier en fonction de l'endroit et en fonction du type de trafic sur lequel la fonction de hachage est appliquée*. Il n'est dès lors pas possible de définir *une* fonction de hachage qui soit bonne en toutes circonstances et pour tous les types de trafic. Dans cette étude, nous nous employons à

[19] c'est-à-dire, la représentation binaire de l'adresse IP source suivie des représentations binaires de l'adresse destination, du port source, du port destination et du protocole.

rechercher, parmi les meilleures méthodes de hachage théoriques[20], celles qui "fonctionnent le mieux" en ce qui concerne le trafic généralement présent sur les réseaux BACKBONE réels.

Nous étudions les méthode de hachage de manière plus approfondie à la section 2.4

2.3.2.2 Méthode de distribution reposant sur l'utilisation de nombres aléatoires

Cette méthode, respectant l'ordonnancement des paquets dans les flux, a été proposée par DAVID THALER et CHINYA RAVISHANKAR [THA98]. Elle associe un *poids* à chaque ligne de sortie, ce poids étant calculé au moyen d'une *fonction pseudo-aléatoire simple* "alimentée" avec l'identificateur du flux et l'identificateur du NEXT-HOP. Quand un paquet arrive, les poids sont générés et le NEXT-HOP recevant le plus grand poids est utilisé pour acheminer le paquet. Nous ne testerons pas cette méthode dans le cadre de cette étude.

2.4 Les méthodes de hachage

Nous avons vu à la section 2.3.2.1 que les méthodes de hachage sont de bonnes méthodes de répartition *pour un trafic uniformément distribué.* Nous les examinons de manière plus approfondie dans cette section.

2.4.1 Hachage suivant l'adresse de destination

2.4.1.1 Principe

Cette fonction de hachage calcule le reste de la division de l'adresse IP destination par le nombre de lignes de sortie. Elle est de la forme :

$h = (x \bmod n)$ où - x représente les 32 bits de la représentation binaire de l'adresse IP de destination
- n est le nombre de lignes de sortie.

D'après [JAI92], une partie seulement des bits constituant l'adresse de destination peut être prise en considération. Si le nombre de lignes est égal à $n = 2^b$, par exemple, il n'est nécessaire de prendre en considération que les b bits de la fin de l'adresse de destination. Ce sont effectivement eux qui sont susceptibles de fournir l'information la plus significative et donc de fournir la distribution de trafic la plus optimale. Nous tenons compte de cette optimisation dans notre algorithme.

Un exemple de *hachage suivant l'adresse de destination* est illustré à la figure 2.7 pour $n = 4$ et pour les paquets *a, b, c, d, e* et *f* d'adresses respectives 192.168.2.12, 192.168.4.4, 192.164.48.13, 192.168.11 et 192.168.2.14.

[20] C-à-d les meilleures méthodes de hachage lorsque les clés sont uniformément distribuées.

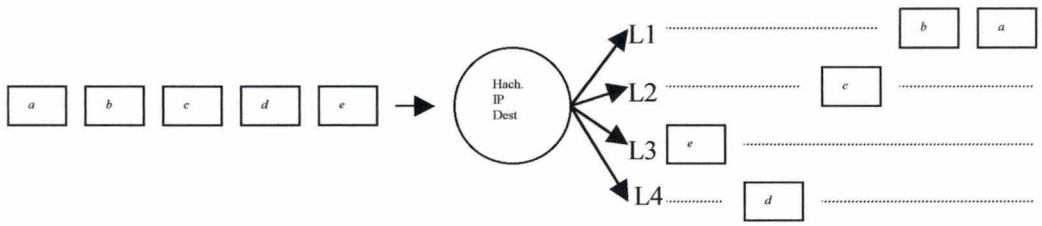


Figure 2.7: Exemple de hachage IP

2.4.1.2 Avantages et inconvénients

La méthode de hachage IP destination présente les avantages suivants :

- il s'agit d'une excellente méthode de hachage (cfr. section 2.3.2.1) pour des adresses IP destination uniformément distribuées
- elle demande peu de calculs, ce qui est important puisqu'elle doit fonctionner dans les routeurs à hauts débits
- elle est facilement implémentable, aussi bien en hardware qu'en software.
- elle est de plus déjà implémentée dans la plupart des routeurs.

Rien ne prouve cependant à priori que la répartition du trafic soit aussi performante dans les réseaux BACKBONE réels, où le trafic n'est pas uniformément distribué. Un trafic non uniformément distribué risque par exemple d'entraîner une parité non uniformément distribuée, qui risque à son tour d'entraîner une mauvaise répartition des paquets entre les différentes lignes de sortie[21]. Il nous reste donc à vérifier expérimentalement si cette méthode répartit bien le trafic dans le cas des réseaux BACKBONE réels.

2.4.2 Hachage utilisant XOR sur l'adresse de destination

2.4.2.1 Principe

Cette méthode utilise également les 32 bits de l'adresse de destination. Elle "hache" l'adresse de destination en quatre morceaux de 8 bits et applique ensuite l'opérateur logique XOR sur les différents morceaux. L'opérateur XOR ("ou exclusif") est préféré à l'opérateur "addition" parce qu'il évite les débordements et les retenues. Il est également préféré aux opérateurs "et" et "ou" parce qu'il ne rend pas systématiquement un nombre plus petit (cas du "et") ou plus grand (cas du "ou") que ses arguments, ce qui risque de provoquer l'envoi d'un plus grand nombre de paquets vers certaines lignes. La fonction de hachage est de la forme :

$$h = (D_1 \oplus D_2 \oplus D_3 \oplus D_4) \bmod n$$

[21] Dans le cas où le nombre n de lignes est pair, par exemple, tous les paquets de clé paire vont être dirigés vers les lignes paires et tous les paquets de clé impaire vont être dirigés vers les lignes impaires. Les paquets risquent donc fort d'être très mal répartis entre les différentes lignes de sortie.

où D_i est le ième octet de l'adresse IP de destination.

Exemple: h(192.168.2.4) = 11000000 XOR 10101000 XOR 00000010 XOR 00000100 = 01101110

2.4.2.2 Avantages et inconvénients

La fonction de hachage XOR IP DESTINATION est très facile à implémenter dans les routeurs. Il s'agit d'une excellente fonction de hachage dans le cas où le trafic est uniformément distribué (cfr. section 2.3.2.1). Le problème lié à l'uniformité de la parité devrait être atténué grâce à l'application de l'opérateur XOR.

En ce qui concerne les réseaux BACKBONE réels, certains problèmes sont cependant susceptibles de se produire. Les permutations d'une même adresse IP sont par exemple toutes hachées de la même façon[22].

2.4.3 Hachage utilisant XOR sur l'adresse IP source et l'adresse IP destination

2.4.3.1 Principe

Le principe général de cette méthode est le même que dans la méthode précédente.
La fonction de hachage est de la forme :

$$h = (S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4) \bmod n$$

où S_i et D_i sont respectivement les ième octets de l'adresse IP source et de l'adresse IP destination.

2.4.3.2 Avantages et inconvénients

Cette fonction devrait au moins aussi bien distribuer le trafic que la fonction précédente. Probablement même mieux, puisqu'un plus grand nombre de bits du flux (deux fois plus) sont pris en considération.

---

[22] Par exemple: h(192.168.2.4) = h(192.168.4.2).

2.4.4 INTERNET CHECKSUM

Cette somme de contrôle est celle utilisée pour calculer les sommes de contrôle des paquets IP, TCP et UDP. Elle est décrite dans le RFC 791 [POS81-a].

2.4.4.1 Principe de la méthode décrite dans le RFC 791

On commence par mettre la somme de contrôle à zéro. Puis, en considérant la totalité de l'en-tête du paquet comme une suite d'entiers de 16 bits, on fait la somme de ces entiers *en complément à 1*. On complémente à 1 cette somme et cela donne le total de contrôle que l'on insère dans le champ CHECKSUM prévu. A la réception du datagramme, on additionne tous les nombres de l'en-tête et si l'on obtient un nombre avec tous ses bits à 1, on suppose que la transmission s'est passée sans problème. Le code C de l'algorithme est montré en annexe 1.

2.4.4.2 Application au LOAD BALANCING

L'algorithme décrit ci-dessus ne peut être repris tel quel dans le cadre du LOAD BALANCING. Il prend effectivement comme *clé* l'ensemble des bits contenus dans l'en-tête du paquet IP. L'algorithme TCP CHECKSUM [POS81-b], par exemple, effectue ses calculs en prenant comme arguments l'ensemble des champs de l'en-tête TCP ainsi que les champs adresse source, adresse destination, protocole et la longueur du paquet TCP (qui ne se trouve pas explicitement transmise mais qui peut être calculée). Deux paquets issus d'un même flux peuvent donc très bien fournir des valeurs de CHECKSUM différentes et par conséquent être dirigés vers des lignes de sortie différentes. Nous devons donc faire en sorte que l'algorithme CHECKSUM ne prenne en compte que les bits appartenant aux cinq champs caractérisant le flux.

La fonction de hachage s'écrit sous la forme suivante :

$$h = \textit{somme de contrôle (adresse IP source, adresse IP destination, port source, port destination, protocole)} \ \bmod \ n$$

2.4.4.3 Avantages et inconvénients

L'algorithme INTERNET CHECKSUM est un peu plus complexe que les méthodes de hachage décrites aux sections précédentes. Il présente bien entendu l'avantage d'être déjà implémenté dans les routeurs. Cet avantage ne permet cependant aucune économie de calcul : l'algorithme INTERNET CHECKSUM implémenté dans les routeurs calcule[23] la somme de contrôle en prenant, en principe[24], comme clé l'ensemble des bits compris dans l'entête des paquets IP. La valeur de la somme de contrôle obtenue ne peut donc être utilisée pour effectuer le LOAD BALANCING[25]. L'algorithme INTERNET CHECKSUM doit donc s'exécuter une deuxième fois dans le routeur - ce qui est un inconvénient majeur puisque les routeurs BACKBONE INTERNET fonctionnent à hauts débits. On peut néanmoins imaginer d'implémenter l'algorithme INTERNET CHECKSUM en parallèle avec l'algorithme de routage, comme expliqué pour la méthode de hachage CRC à la section 3.3.5.

Dans leur article [STO98], JONATHAN STONE et MICHAEL GREENWALD démontrent (théorème 6 en appendice A) que l'algorithme TCP CHECKSUM fournit une répartition uniforme des valeurs des sommes de contrôle si, du moins, les valeurs de la clé sont *uniformément distribuées.* Rien ne prouve cependant à priori que ce soit également le cas en ce qui concerne les réseaux BACKBONE réels, où le trafic n'est pas uniformément distribué.

La méthode de hachage INTERNET CHECKSUM devrait, en principe, être plus performante que les méthodes de hachage décrites aux paragaphes précedents: la clé utilisée comprend en effet un plus grand nombre de bits (les 104 bits caractérisant le flux sont ici pris en considération) et il ne semble plus y avoir de problèmes liés aux permutations, à l'uniformité des parités, etc. L'algorithme ne doit pas prendre en considération le fait que la machine utilisée soit BIG ENDIAN ou LITTLE ENDIAN.

2.4.5 Somme de contrôle CRC 16

La méthode CRC (CYCLIC REDUNDANCY CHECK) est une méthode de détection d'erreurs utilisée au niveau de la couche 2 (couche liaison de données).
Dans le cadre du LOAD BALANCING, nous nous intéressons bien entendu uniquement à la partie "calcul de la somme de contrôle" de l'algorithme CRC. La partie "détection de l'erreur" ne nous intéressant évidemment pas.

---

[23] La somme de contrôle doit être recalculée lors du passage d'un paquet IP dans un routeur puisqu'il y a modification d'au moins un champ du paquet: il s'agit du TTL, qui est décrémenté d'une unité à chaque passage dans un routeur.

[24] Il se pourrait que cette somme de contrôle ne soit pas recalculée entièrement. Certaines optimisations permettent effectivement de calculer la nouvelle somme de contrôle en fonction de la somme de contrôle calculée par le routeur précédent. On peut imaginer également que, pour des raisons de performance, les routeurs du réseau BACKBONE ne vérifient pas systématiquement l'intégrité des paquets IP qui leur arrivent. Cette vérification est d'ailleurs inutile dans le cas où les routeurs font du switching plutôt que du routing. Elle est même inutile dans le cas où les routeurs font du routing optique (dans ce cas, ils fonctionnent effectivement comme des switchs).

[25] Rappelons que la méthode de hachage INTERNET CHECKSUM ne doit s'appliquer ici que sur les cinq champs caractérisant le flux.

2.4.5.1 Principe de la méthode

Cette méthode utilise un polynôme générateur *g(x)*, qui dépend uniquement de l'implémentation CRC utilisée[26]. Soit *r*, le degré de *g(x)*. Les 104 bits du flux sont utilisés pour former un polynôme:

$F(x) = d_{103}X^{103} + d_{102}X^{102} + d_{102}X^{101} + \ldots + d_2X^2 + d_1X^1 + d_0X^0$.

Ce polynôme *F(x)* est ensuite multiplié par X^r pour former le polynôme *d(x)*.
Le polynôme *d(x)* est ensuite divisé (modulo 2) par le polynôme fixe *g(x)*. On obtient alors un polynôme *r(x)*. Les bits de la somme de contrôle CRC sont les coefficients de ce polynôme *r(x)*. Le choix de *g(x)* donne la taille de la somme de contrôle CRC. Dans le cas où *g(x)* est un polynôme de degré 16,

$d(x) = X^{16}\ d_{103}X^{119} + d_{102}X^{118} + d_{102}X^{117} + \ldots + d_2X^{18} + d_1X^{17} + d_0X^{16}$

et

$r(x) = b_{15}X^{15} + b_{14}X^{14} + \ldots + b_1X + b_0$

de degré au plus égal à 15. La somme de contrôle est donc sur 16 bits et il y a 2^{16} sommes de contrôle possibles.

On parle de CRC 16 pour des sommes de contrôle sur 16 bits, de CRC 32 pour des sommes de contrôle sur 32 bits, ...

2.4.5.2 Avantages et inconvénients

Cette méthode de hachage est évidemment plus complexe que les méthodes de hachage précédentes. L'étude [BRO61] montre cependant qu'un simple registre à décalage suffit pour obtenir la somme de contrôle. On pourrait également faire en sorte que l'algorithme CRC s'exécute parallèlement à l'algorithme de routage. Ainsi, une fois l'algorithme de routage terminé, la somme de contrôle calculée en parallèle serait utilisée pour calculer un numéro de ligne de sortie. Ceci entraîne bien sûr un calcul inutile de la somme de contrôle dans le cas où l'algorithme de routage ne sélectionnerait qu'une seule ligne de sortie mais peu importe puisque cela n'entraîne de toute manière aucune perte de temps dans les routeurs.

La méthode CRC ne pose donc aucun problème réel en ce qui concerne son implémentation dans les routeurs à hauts-débits des BACKBONE INTERNET. Cette méthode est de plus une excellente méthode d'un point de vue théorique : si la répartition du trafic (et donc des clés) était complètement uniforme, la probabilité que deux clés aient la même somme de contrôle serait de $1/2^{16}$ pour CRC16 et de $1/2^{32}$ pour CRC32. RAJ JAIN a trouvé de bons résultats pour la

[26] En ce qui concerne *g(x)*, les deux polynômes de degré 16 normalisés sont:
- $x^{16} + x^{12} + x^5 + 1$ (standart x25)
- $x^{16} + x^{15} + x^2 + 2$ (CRC16).

Le polynôme de degré 32 le plus répandu est $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$. C'est ce polynôme qui est utilisé pour le calcul du CRC des trames ETHERNET

fonction CRC32 dans son étude [JAI92] concernant une trace prise sur un réseau ETHERNET. Les résultats de JAIN peuvent laisser croire que des résultats similaires pourraient être obtenus dans un réseau IP. N'oublions cependant pas que le contexte est ici très différent puisque nous travaillons dans les réseaux BACKBONE. Les choses pourraient donc se passer très différemment.

2.5 Conclusions

D'un point de vue expérimental, il est possible que certaines de ces méthodes soient plus performantes que les autres : l'homogénéité de l'ensemble des clés de hachage n'est en effet pas connue dans les réseaux BACKBONE réels et il est donc très difficile de se faire *à priori* une idée sur la manière dont les différentes méthodes de hachage vont répartir le trafic. Nous devons donc vérifier *expérimentalement* les performances des différentes méthodes de hachage. Nous nous y employons au chapitre 4.

Chapitre 3

Simulation du trafic INTERNET BACKBONE

3.1 Introduction

Nous commençons par expliquer les raisons pour lesquelles nos algorithmes de hachage sont simulés sur du trafic réel plutôt qu'implémentés directement dans un routeur. Nous expliquons ensuite la différence entre les logiciels de type TCPDUMP [DUM01] et le logiciel NETFLOW [NET99]. Enfin, nous discutons des avantages et inconvénients respectifs de ces logiciels en ce qui concerne les méthodes de répartition de trafic.

3.2 Pourquoi faut-il simuler ?

Il n'est pas possible d'appliquer directement nos algorithmes de hachage sur le trafic des principales lignes des réseaux BACKBONE :

- nous ne sommes pas gestionnaire d'un réseau BACKBONE et n'avons donc pas la permission de configurer nous-mêmes un routeur BACKBONE.
- Nous n'avons de toute manière pas la maîtrise ni le savoir-faire pour implémenter nous-mêmes nos algorithmes dans un routeur.
- Nos algorithmes présentent une certaine complexité et ne sont pas implémentés en HARDWARE. Ils peuvent donc difficilement suivre le débit élevé des lignes des réseaux BACKBONE.
- Les performances des différentes méthodes de hachage doivent être évaluées simultanément, ce qui rend le problème encore plus complexe.

Il semble raisonnable de procéder de la manière suivante :

- demander à un gestionnaire de réseau BACKBONE de bien vouloir capter et stocker du trafic à notre place
- appliquer ensuite nos différents algorithmes de hachage sur la trace fournie par ce gestionnaire.

Il est cependant difficile d'obtenir ce type de trace. Les gestionnaires de trafic préfèrent en effet ne pas trop dévoiler les caractéristiques du trafic qui passe par leur réseau et cherche à protéger les données jugées confidentielles par eux ou par leurs clients.

3.3 Les logiciels de capture de trafic

3.3.1 Les logiciels de type TCPDUMP

Un logiciel de capture de type TCPDUMP [JAC92] [DUM01] ne s'exécute pas sur un routeur mais sur une machine distante connectée à une ligne quelconque d'entrée ou de sortie d'un

routeur. Il capture tous les paquets circulant sur la ligne d'entrée ou de sortie. Il stocke le temps de capture de ces paquets, une grosse partie de leur entête (adresse IP source, adresse IP destination, ...) et, éventuellement, leur contenu. La figure 3.1 ci-dessous nous montre un exemple de trace TCPDUMP.

```
11:08:38.678022 123.44.21.123.1183 > 115.61.90.51.80: S 184624:184624(0)
win 8192 <mss 1460> (DF)
                  4500 002c 910d 4000 7f06 aba8 8a30 166d
                  c344 5b34 049f 0050 0002 d130 0000 0000
                  6002 2000 e2ee 0000 0204 05b4 05b4
11:08:38.684533 131.182.173.82.6699 > 118.32.66.15.2876: P
78167246:78167286(40) ack 967526090 win 8679 (DF) [tos 0xc0]
                  45c0 0050 5020 4000 7006 c9ce 81ba b356
                  8a30 30b8 1a2b 0b3c 04a8 bcce 39ab 46ca
                  5018 21e7 ecde 0000 08d5 6b2e b4f1 4e76
                  2b20 9a97 b953
11:08:38.696346 95.31.8.63.137 > 123.18.22.201.137: udp 50
                  4500 004e 0393 0000 8011 17bc 8a30 04f1
                  8a30 05ff 0089 0089 003a c728 8273 0110
                  0001 0000 0000 0000 2045 4a45 4f46 4446
                  4545 4245 4d45
11:08:38.699913 124.44.21.110.1180 > 185.61.90.50.80: F 184963:184963(0)
ack 915108164 win 8311 (DF)
                  4500 0028 900d 4000 7f06 acac 8a30 166d
                  c344 5b34 049c 0050 0002 d283 368b 7144
                  5011 2077 5105 0000 0000 0000 0000
11:08:38.701307 123.13.45.23.1140 > 100.13.52.128.4670: . ack 36935556 win
8576 (DF)
                  4500 0028 cd1f 4000 7f06 ac28 8a30 195d
                  c835 16c5 0474 123e 0088 f508 0233 9784
                  5010 2180 65d2 0000 0000 0000 0000
```

Figure 3.1:Exemple de trace TCPDUMP

La trace montrée à la figure 3.1 comprend quatre paquets. Nous avons anonymisé les adresses IP des paquets dans le but de garantir la confidentialité. A titre d'exemple, nous décrivons le premier paquet. TCPDUMP affiche tout d'abord l'estampille horaire (le TIMESTAMP) dans le format HH:MM:SS.µS (*11:08:38.678022*). TCPDUMP affiche ensuite l'adresse IP et le port de la machine émettrice (*123.44.21.123.1183*), l'adresse IP et le port de la machine destination (*115.61.90.51.80*) et le flag (*S* signifie le début d'une connexion TCP). Le champ *184624:184624(0)* signifie que le numéro de séquence du paquet est *184624* et le nombre d'octets de données contenus dans le paquet est 0. Le champ *win 8192* indique la taille de fenêtre réclamée par l'émetteur. Le champ *<mss 1460>* indique la taille maximum de segment (MSS) spécifiée par l'émetteur. L'émetteur ne veut pas recevoir de segments plus grands que cette valeur. L'option *-x* de la commande TCPDUMP nous a permis d'afficher le contenu de tout le paquet en hexadécimal.

Les informations étant stockées *paquet par paquet*, il est possible, en relisant la trace obtenue, de simuler à peu près le trafic présent lors de la capture. Les logiciels de capture de type TCPDUMP s'avèrent donc à priori très intéressants dans le cadre de notre étude.

Leur inconvénient majeur réside cependant dans le fait qu'ils obligent à stocker *tous* les paquets. Dans le cas des lignes BACKBONE INTERNET à hauts débits, le nombre de paquets à stocker est très élevé, ce qui nécessite un espace de stockage important.
A titre d'exemple, prenons le cas de la ligne OC-48 (cfr. chapitre 1) SAN FRANCISCO - LOS ANGELES du BACKBONE AT&T (figure 1.1 du chapitre 1). Le débit maximum d'une telle ligne est égal à 2,5 Gbps. Supposons un trafic moyen égal à 1 Gbps, ce qui semble raisonnable puisqu'il correspond à 40% de la capacité maximale de la ligne. L'espace de stockage nécessaire représente au moins une centaine de GByte par heure de trafic.

Nous avons vu au paragraphe 3.1 qu'il était difficile d'obtenir des traces de trafic réelles. Des sites comme http://moat.nlanr.net ou comme http://www.caida.org essaient de remédier partiellement à cet inconvénient en proposant des traces de type TCPDUMP. Ces traces ont été construites à partir de trafic réel et sont fournies dans le but de stimuler la recherche scientifique. Elles comprennent l'entête complète de tous les paquets IP. Il est donc possible, en les relisant à l'aide d'un script, de "rejouer", au moins en partie, le trafic présent lors de la capture. Ces traces présentent cependant un inconvénient important : il s'agit de traces anonymisées, les adresses IP de leurs paquets ayant été modifiées dans le but de garantir la confidentialité. Nous ne connaissons pas la manière dont ces adresses ont été modifiées. Il est fort possible que la distribution de ces adresses ne soit pas identique à la distribution des adresses du trafic réel. L'application des différentes méthodes de hachage sur ces adresses IP source et destination modifiées risque donc d'amener à des résultats différents de ceux qui seraient obtenus à partir de traces de trafic réelles. L'utilisation de telles traces risque donc de nous induire en erreur.

3.3.2 NETFLOW

Une carte de type NETFLOW peut être présente dans un routeur. Cette carte identifie en temps réel les flux[27] des paquets IP [CIS00-b], résume et stocke les informations *flux par flux* dans un cache. Le format des résumés de flux est montré à la figure 3.3. Episodiquement, la carte NETFLOW exporte les *résumés de flux* obtenus de manière asynchrone, dans un simple paquet UDP (figure 3.2), du routeur vers une machine distante.

Notons que la notion de *flux* considérée par NETFLOW ne correspond pas tout à fait à celle définie dans notre étude au chapitre 1. Un *flux* NETFLOW correspond à un ou plusieurs paquets IP utilisant le même protocole IP, les mêmes adresses IP source et destination, les mêmes numéros de port TCP [POS81-b] ou UDP [POS80], le même TOS[28] et le même identificateur d'interface d'entrée [CIS00-a]. Ceci ne pose cependant pas de problème particulier en ce qui concerne notre étude. Les données fournies par la carte NETFLOW peuvent en effet être agrégées de manière à obtenir des flux tels que définis au chapitre 1.

---

[27] Nous avons défini la notion de *flux* au chapitre 2. Un *flux* correspond à un ou plusieurs paquets IP utilisant le même protocole IP, les mêmes adresses source et destination et les mêmes numéros de ports TCP ou UDP.

[28] Type de service. Il s'agit ici des bits se trouvant dans le champ TOS de l'en-tête d'un paquet IP.

Le logiciel de gestion du cache NETFLOW contient une batterie d'algorithmes [CIS00-a]. Ceux-ci:

- déterminent si un paquet fait partie d'un flux déjà présent dans le cache ou s'il faut générer une nouvelle entrée dans le cache,
- mettent à jour dynamiquement les données concernant les différents flux présents dans le cache
- tiennent compte des règles d'expiration des caches. Un flux peut être exporté (et donc retiré du cache NETFLOW du routeur) pour une des raisons suivantes [CIS00-a] :

a) le cache est plein et un nouveau flux doit être créé.

b) un paquet signalant la fin d'un flux est reçu (exemple : un paquet contenant un segment TCP avec le bit FIN mis à 1).

c) un flux est présent depuis trop longtemps dans le cache.

d) plus aucun paquet n'a été reçu par un flux depuis un certain temps.

Quand des informations concernant un flux sont prêtes à être exportées, NETFLOW essaie d'envoyer d'autres informations concernant d'autres flux [LEI99]. Chaque résumé de flux a une certaine longueur et plusieurs résumés de flux peuvent être envoyés à la fois dans un simple paquet UDP [CIS00] (figure 3.2). Dans la version NETFLOW V5, les résumés de flux ont une longueur de 48 bytes et on peut en envoyer jusqu'à 30 à la fois dans un simple paquet UDP.

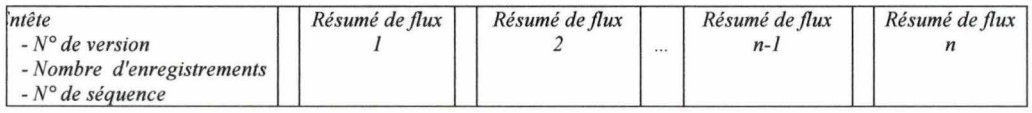


Figure 3.2 : Structure d'un paquet UDP exporté par NETFLOW.

Chaque *en-tête* de paquet UDP contient au moins :

- le numéro de version (par exemple *5*, dans le cas de la version NETFLOW v5). La plupart des applications réceptrices s'adaptent à la version NETFLOW indiquée dans l'entête en allouant automatiquement un espace mémoire susceptible de stocker le plus grand datagramme UDP possible pour cette version.
- le nombre d'enregistrements dans le datagramme.

Les entêtes des paquets UDP des versions 5, 7 et 8 comprennent également un numéro de séquence, ce qui permet de détecter les paquets perdus. Il n'y a cependant pas de possibilité de retransmission des paquets perdus.

Dans le cas de NETFLOW v5, chaque résumé de flux a le format suivant (figure 3.3)[CIS99] :

Bytes	*Contenu*	*Description*
0-3	Srcaddr	Adresse IP source
4-7	Dstaddr	Adresse IP destination
8-11	Nexthop	Adresse IP du next hop router
12-13	Input	Index de l'interface physique d'entrée
14-15	Output	Index de l'interface physique de sortie
16-19	DPkts	Nombre de paquets comptés pour le flux
20-23	DOctets	Nombre de bytes comptés pour le flux
24-27	First	Temps auquel le premier paquet du flux est reçu
28-31	Last	Temps auquel le deuxième paquet du flux est reçu
32-33	Srcport	Numéro du port source TCP/UDP ou équivalent
34-35	Dstport	Numéro du port destination TCP/UDP ou équivalent
36	Pad 1	Lien vers des informations supplémentaires. En pratique, ce byte est inutilisé.
37	tcp_flags	Flags TCP (OR cumulatif des flags TCP)
38	Prot	Protocole IP
39	Tos	Type de service (ToS)
40-41	Src_as	Numéro du Système Autonome source
42-43	Dst_as	Numéro du Système Autonome destination
44	Src_mask	Bits de préfixe du masque de l'adresse source
45	Dst_mask	Bits de préfixe du masque de l'adresse destination
46-47	Pad2	Lien vers des informations supplémentaires. En pratique, ce byte est inutilisé.

Figure 3.3: Structure d'un résumé de flux NETFLOW v5.

Seuls les champs *adresse* IP *source, adresse* IP *destination, numéro du port source, numéro du port destination et protocole* IP nous intéressent bien entendu dans le cadre de notre étude.

Les données produites par les cartes NETFLOW peuvent être recueillies sur une machine distante par un logiciel de collecte comme, par exemple, CISCO NETFLOW FLOW COLLECTOR *v 3* (figure 3. 4) [CIS00-c] , CAIDA CFOWD [CFL01], FLOWSCAN [PLON00] ou FLOWTOOLS [ROM00].

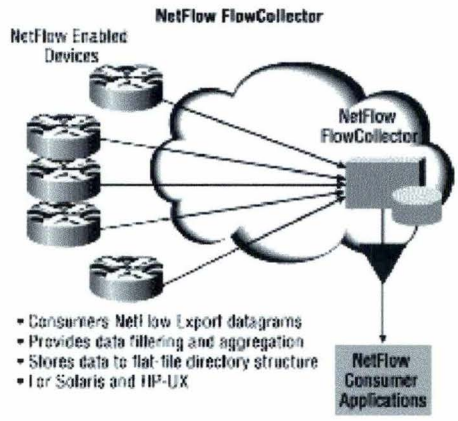


Figure 3.4 : Schéma de principe de CISCO NETFLOW COLLECTOR v3

Ces logiciels stockent les données reçues dans des fichiers selon un format qui leur est propre.

Les logiciels CISCO NETFLOW FLOW COLLECTOR *v 3*, CAIDA CFOWD et FLOWSCAN permettent de réduire le volume des données (filtrage, agrégation) et de gérer leur stockage. CISCO NETFLOW FLOW COLLECTOR v 3 permet par exemple d'agréger les données de manière à obtenir des flux conformes à notre définition du chapitre 1 [CIS00-c].

Le logiciel FLOWTOOLS se contente simplement de capter les paquets UDP. Il ne filtre pas et n'agrège pas. Il est donc moins complexe que les autres et fournit des données brutes (c'est à dire telles qu'elles sont exportées par NETFLOW). Ceci le rend très intéressant à utiliser dans le cadre de notre étude :

- les routeurs BACKBONE sont susceptibles d'être traversés par un très grand nombre de flux en même temps. La plupart de ces flux ont une durée de vie relativement courte. La variabilité des flux présents dans le routeur risque donc d'être relativement importante. La carte NETFLOW risque dès lors d'exporter les résumés de flux à fréquence élevée. Le logiciel de capture présent sur la machine distante va recevoir les paquets UDP à cette fréquence. Il est fort possible que les logiciels CISCO NETFLOW FLOW COLLECTOR *v 3*, CAIDA CFOWD [CFL01] et FLOWSCAN [PLON00], vu leur complexité importante, soient incapables de suivre une fréquence aussi élevée d'arrivée de paquets UDP. Ils risquent dès lors de ne pas capter tous les paquets et de fausser ainsi les résultats obtenus. Le logiciel FLOWTOOLS étant, par contre, beaucoup plus simple, risque beaucoup moins de perdre des paquets. Il est donc nettement plus fiable.
- FLOWTOOLS fournit des données brutes, donc plus fiables.

3.4 Avantages et inconvénients en ce qui concerne le LOAD BALANCING

Les logiciels de type TCPDUMP offrent l'*avantage* de fournir des résultats presque[29] identiques à ceux qui seraient obtenus par implémentation directe dans un routeur. L'application de nos algorithmes de répartition de trafic sur une trace de type TCPDUMP devrait donc amener des résultats fiables, nous permettant d'évaluer de manière précise les performances des différentes méthodes de répartition de trafic.
L'*inconvénient* majeur des logiciels de type TCPDUMP réside cependant dans le fait qu'ils obligent à stocker tous les paquets. Dans le cas de lignes BACKBONE INTERNET à hauts débits, le nombre de paquets à stocker est très élevé, ce qui nécessite un espace de stockage important. Ceci oblige à effectuer les mesures expérimentales sur des intervalles de temps relativement courts.

Un fichier de type NETFLOW ne comprend pas des informations *paquet par paquet* mais des informations *flux par flux*. Le fait de stocker l'information sous la forme de résumés de flux présente évidemment le gros *avantage* d'économiser l'espace de stockage - ce qui est de nature à permettre des captures et des stockages de trafic sur des intervalles de temps plus élevés que ceux possibles avec TCPDUMP. NETFLOW présente cependant un certain nombre d'*inconvénients*. Un premier inconvénient de NETFLOW réside dans le fait qu'il peut charger le CPU du routeur qui exporte les informations. Un autre inconvénient est qu'il est moins précis que TCPDUMP, les informations étant stockées flux par flux. Avec NETFLOW, il n'est effectivement pas possible de retrouver *toute* l'information concernant le trafic réel. NETFLOW ne fait qu'approximer le trafic réel. L'application de nos algorithmes sur une trace NETFLOW *risque* donc de fournir des résultats différents de ceux qui seraient obtenus avec du trafic réel. Ceci *risque* bien sûr d'amener à des conclusions erronées en ce qui concerne l'évaluation des performances des différentes méthodes de hachage.

Au chapitre 4, nous essayons de déterminer s'il y a réellement un risque d'utiliser NETFLOW à la place de TCPDUMP. Nous mesurons l'erreur effectuée en NETFLOW plutôt que TCPDUMP. Si cette erreur s'avère être faible, nous pourrons appliquer nos algorithmes sur des traces de type NETFLOW. Nous pourrons ainsi simuler les différentes méthodes de répartition de trafic sur des intervalles de temps beaucoup plus grands.

---

[29] *Presque identiques* car la manière dont le timestamp de TCPDUMP est établi et les mécanismes internes aux routeurs sont un peu flous.

Chapitre 4

Etude expérimentale

Nous étudions, *dans une première partie*, les caractéristiques de la trace utilisée. Il s'agit d'une trace anonymisée[30], étant donné la difficulté d'obtenir des traces de trafic réelles de la part des gestionnaires de réseau. La trace anonymisée est une trace TSH (trace de type TCPDUMP) issue d'une grande université américaine [NLA1].

Dans une deuxième partie, nous *simulons*[31] les différentes méthodes de répartition de trafic sur la trace TSH et interprétons les résultats obtenus.

Dans une troisième partie, nous *transformons* la trace TSH en une trace NETFLOW. Nous simulons ensuite les différentes méthodes de répartition de trafic sur la trace NETFLOW obtenue. Les résultats obtenus sont comparés avec ceux de la première partie de ce chapitre. Nous mesurons l'*erreur* effectuée en utilisant la trace NETFLOW plutôt que la trace TSH. Nous interprétons l'erreur obtenue.

4.1 Caractéristiques de la trace

La trace utilisée correspond au trafic capté pendant un intervalle de temps de +/- 90 s sur une ligne 35 Mbps d'accès Internet de l'université d'état du COLORADO. Le trafic capté comprend +/- 1,5 millions de paquets issus de +/- 40.000 *flux*[32]. Nous avons constaté des "effets de bords" au début et à la fin de la trace. Ils ne peuvent provenir que de la manière dont la trace a été capturée. Nous montrons l'effet de bord de la fin de la trace à la figure 4.1.

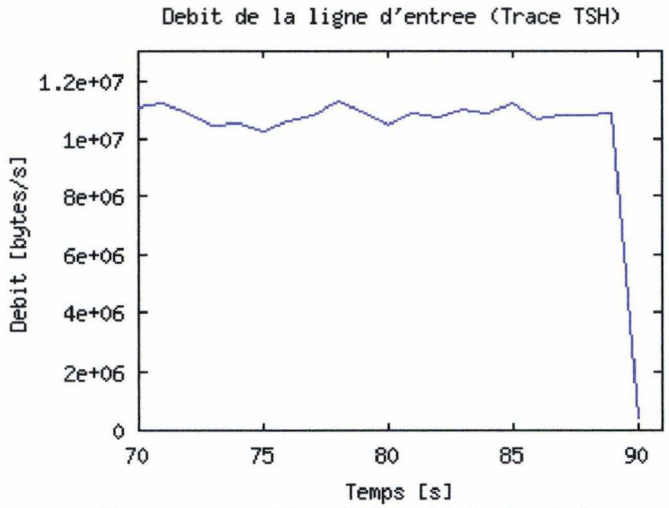


Figure 4.1 : Effet de bord à la fin de la trace.

[30] Une trace anonymisée est une trace dont on a anonymisé les adresses IP sources et destinations dans le but de protéger les données à caractère privé.

[31] Nous expliquons à la section 3.2 pourquoi nous devons simuler.

[32] Nous avons défini la notion de *flux* au chapitre 2. Un *flux* correspond à un ou plusieurs paquets IP utilisant le même protocole IP, les mêmes adresses source et destination et les mêmes numéros de ports TCP ou UDP.

Cette figure montre la répartition du trafic, durant l'intervalle de temps [70, 90] secondes, *avant* application des méthodes de hachage. L'effet de bord apparaît durant l'intervalle de temps [89, 90] secondes.
Nous avons donc effectué nos mesures durant un intervalle de temps plus restreint. Par mesure de précaution, nous avons choisi l'intervalle de temps [5, 85] secondes.

4.1.1 Distribution cumulative des paquets

La distribution cumulative des paquets en fonction de leur taille est montrée à la figure *4.2.* Nous utilisons un graphique *cumulatif* parce qu'il montre mieux l'importance relative, par rapport au nombre total de paquets, du nombre de paquets obtenus pour chaque taille de paquet.

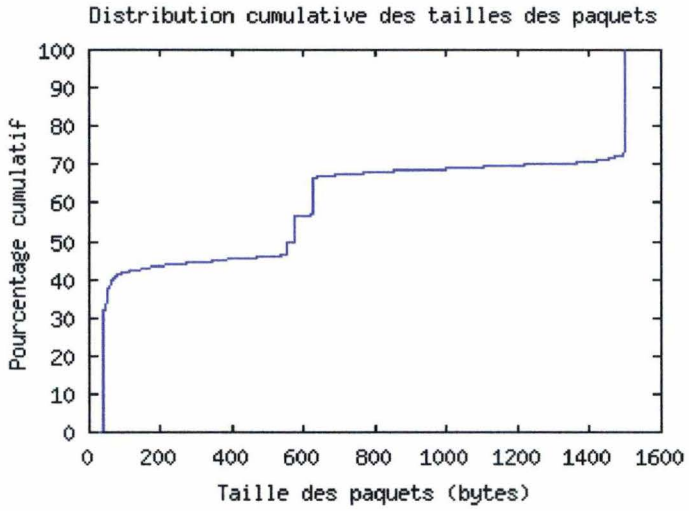


Figure 4.2 : Distribution cumulative des paquets en fonction de leur taille.

Il n'y a pas de paquet de taille supérieure à 1500 bytes. Les tailles des paquets obtenues correspondent pour la plupart aux tailles des paquets IP les plus couramment observées dans les gros réseaux BACKBONE INTERNET[33]. Les paquets de 40-44 bytes (+/- 40% du nombre total de paquets) correspondent aux paquets renfermant les nombreux acquis TCP, aux paquets de contrôle SYN, FIN et RST ainsi qu'aux paquets comprenant les caractères simples frappés au clavier lors d'une session TELNET.
Les paquets de 552 et de 576 bytes (+/- 25% du nombre total de paquets) correspondent à des paquets de 552 et de 576 bytes renfermant chacun un segment de taille maximum (MSS) [BRA89], c'est-à-dire un segment de 512 ou de 536 bytes.
Les paquets de 1500 bytes (+/- 30% du nombre total de paquets) correspondent au MTU[34] des stations connectées à un réseau ETHERNET.

[33] L'étude de GREG MULLER et de KEVIN THOMPSON [THOM98] réalisée en avril 1998 montre la prédominance des petits paquets et des pics pour des tailles de 44, 552, 576 et 1500 bytes.

[34] MAXIMUM TRANSMISSION UNIT. Le MTU est défini comme la capacité de transmission maximale d'une interface. L'encapsulation ETHERNET, par exemple, ne tolère pas des trames de plus de 1500 bytes. Les MTU sont définis dans le RFC 1191 [MO90].

4.1.2 Distribution cumulative des rafales

Nous essayons ici de déterminer si la trace TSH utilisée comporte un grand nombre de rafales (suite de paquets consécutifs issus d'un même flux). Ceci peut effectivement influencer la répartition *instantanée* du trafic sur les différentes lignes de sortie, après application des méthodes de LOAD BALANCING. La distribution cumulative des rafales est montrée à la figure 4.3. Les rafales dépassant 1500 bytes correspondent essentiellement à des rafales constituées de deux paquets (un paquet de 1500 bytes et un paquet de 40, 552, 576 ou de 1500 bytes).

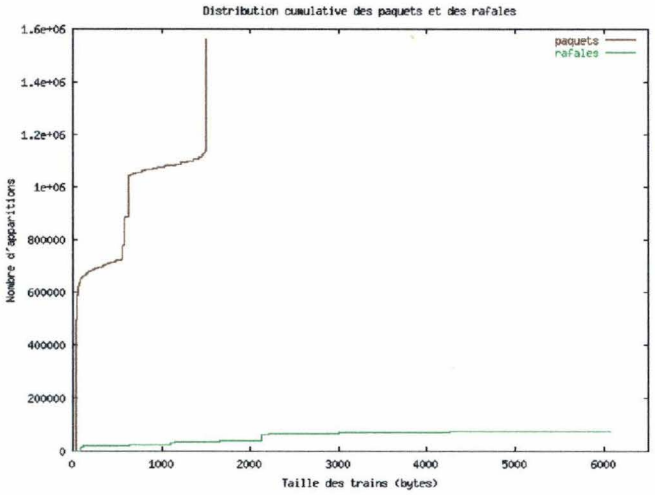


Figure 4.3:Distribution cumulative des rafales en fonction de leur taille.

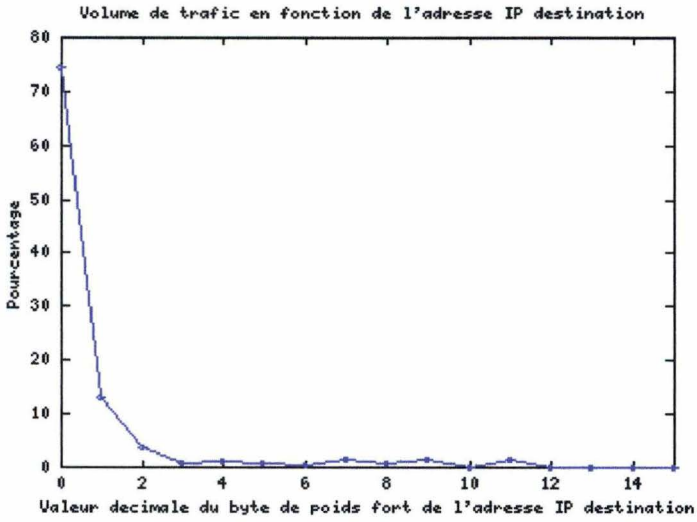
Pour faciliter les comparaisons, nous représentons la distribution cumulative des paquets sur la même figure. Nous constatons que le nombre de rafales apparues est très petit par rapport au nombre de paquets.

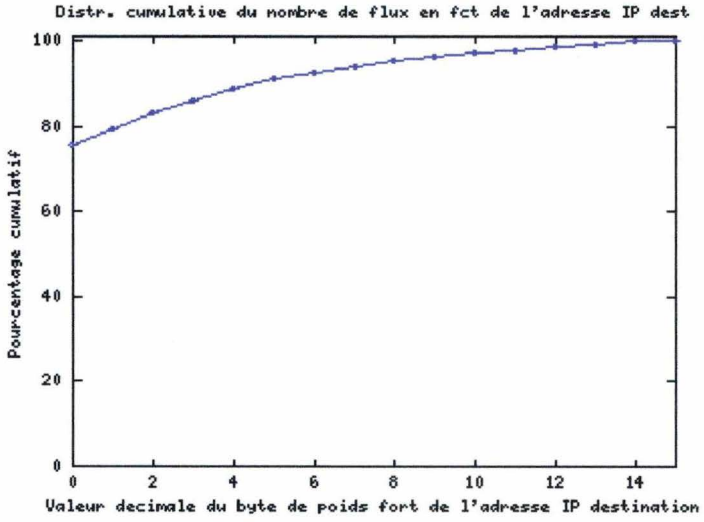
Le volume des rafales est également faible : il ne représente qu'à peu près 4,5 % du volume total du trafic. Le nombre et le volume des rafales étant peu élevés, elles ne devraient pas avoir d'influence négative sur la répartition *instantanée* du trafic sur les différentes lignes de sortie.

4.1.3 Distribution du volume du trafic et du nombre de flux en fonction des adresses IP

Les paquets IP présents dans la trace TSH ont-ils souvent les mêmes adresses (ou classes d'adresses) IP ou, au contraire, ont-ils des adresses IP très différentes et relativement bien réparties ?

Cette question nous semble importante puisque nous utilisons des *méthodes de hachage* pour répartir le trafic et que ces méthodes tiennent toutes compte de l'adresse IP dans leur calcul d'attribution de la ligne de sortie. Nous essayons de répondre à la question en calculant la distribution de *volume du trafic* en fonction des adresses IP.
La distribution du volume du trafic *en fonction des 8 bits de poids fort de l'adresse IP de destination* et la distribution du nombre de flux *en fonction des 8 bits de poids fort de l'adresse* IP de destination sont montrées aux figures 4.4 et 4.5.





Les nombres de 0 à 15 correspondent à la valeur décimale du byte de poids fort :
0 correspond aux 8 bits de poids fort 0000 0000
1 correspond aux 8 bits de poids fort 0000 0001
...
15 correspond aux 8 bits de poids fort 0000 1111

Figures 4.4 et 4.5 : Distribution du volume du trafic et distribution cumulative du nombre de flux en fonction des adresses IP.

Il ne s'agit pas de graphiques continus mais de graphiques discrets : les adresses IP destination sont en effet regroupées en fonction de la valeur décimale des 8 bits de poids fort de l'adresse IP de destination et chaque point représenté sur les graphiques correspond à la distribution mesurée pour une classe d'adresses IP particulière (ensemble de paquets ayant les mêmes 8 bits de poids fort). Les différents points obtenus ont étés reliés entre eux pour mieux montrer l'évolution des distributions en fonction des adresses IP. Le volume du trafic n'est pas représenté de manière cumulative pour des raisons de clarté du graphique. Les distributions *en*

fonction des 32 bits de l'adresse IP et les distributions *en fonction des adresses IP SOURCE* donnent des graphiques similaires.

Nous constatons que tous les flux ont les quatre bits de poids fort de leur adresse IP destination égaux à 0000 et que à peu près 80% du volume de trafic et du nombre de flux ont les 8 bits de poids fort de leur adresse IP destination égaux à 0000 0000. *La répartition des adresses IP présentes dans la trace TSH est donc peu uniforme.* Cette répartition non uniforme du volume du trafic peut provenir de l'anonymisation[35] des adresses IP. L'algorithme utilisé pour l'anonymisation des adresses IP se trouve sur le site de NLANR (NATIONAL LABORATORY FOR APPLIED NETWORK RESEARCH) à l'adresse http://moat.nlanr.net/Traces/Bin/tshenc.pl . La répartition non uniforme des adresses IP peut également provenir du trafic originel.

Cette répartition non uniforme des adresses IP peut entraîner des conséquences en ce qui concerne le LOAD BALANCING. La répartition du trafic sur les différentes lignes de sortie risque effectivement de ne pas être uniforme: les méthodes de hachage tiennent en effet compte des adresses IP source et destination de chaque paquet dans leur calcul d'attribution de la ligne de sortie. Le LOAD BALANCING devrait s'effectuer le moins bien pour les méthodes XOR IP DESTINATION, XOR IP SOURCE ET DESTINATION et IP DESTINATION puisque ces méthodes ne tiennent compte que des adresses IP.

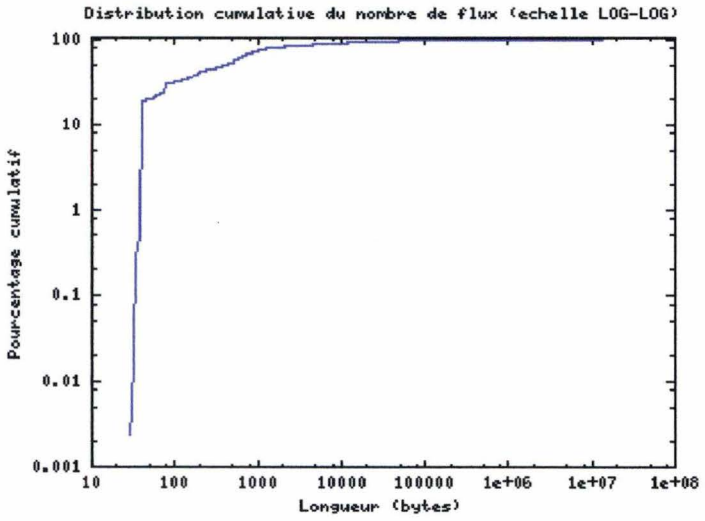
4.1.4 Distribution des flux en fonction de leur longueur

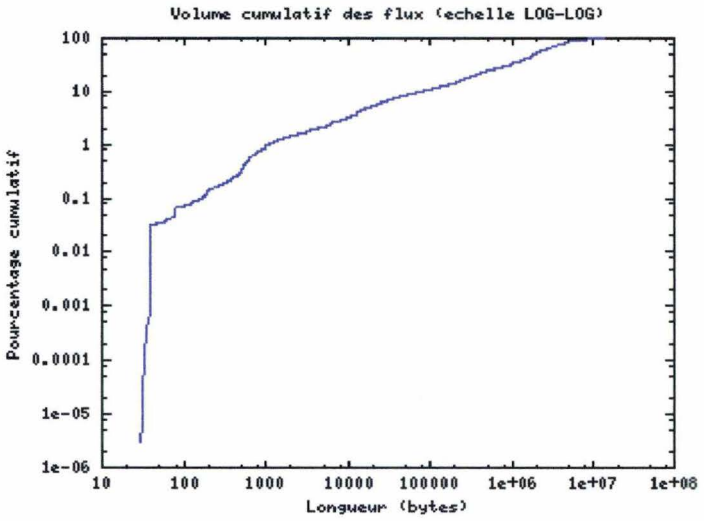
L'essentiel du trafic est-il dû à une multitude de «petits» flux ou, au contraire, à un nombre restreint de «gros» flux ?

Cette question nous semble importante. La probabilité d'obtenir une répartition *uniforme* du trafic sur les différentes lignes de sortie, et donc d'obtenir un "bon" LOAD BALANCING, devrait en effet dépendre de la composition du trafic en terme de "petits" et de "gros" flux. La probabilité d'obtenir une répartition uniformehomogène est très grande dans le cas où le trafic est essentiellement constitué d'un grand nombre de "petits" flux (loi des grands nombres).

La figure 4.6 montre la distribution cumulative du nombre de flux en fonction de leur longueur. La figure 4.7 montre le *volume* cumulatif des flux en fonction de leur longueur. Nous utilisons des graphiques *cumulatifs (échelle* LOG - LOG*)* de manière à mieux montrer l'importance relative, par rapport au trafic total, de la mesure obtenue pour chaque classe d'adresse IP particulière.

[35] Nous avons utilisé une trace anonymisée [NLA1], c'est-à-dire une trace dont on a rendu anonymes les adresses IP sources et destinations dans le but de protéger les données à caractère privé.





Figures 4.6 et 4.7 :
Distribution cumulative du nombre de flux et volume cumulatif des flux en fonction de leur longueur.

Les flux ont des longueurs comprises entre 40 et +/- 10^7 bytes. *La trace est composée d'un grand nombre de "petits" flux* (figure 4.6): 80% des flux ont une longueur inférieure à 1500 bytes et 99,5% des flux ont une longueur inférieure à 10^6 bytes. *Les "petits" flux ne représentent cependant quasiment rien en terme de volume* (figure 4.7) : les flux inférieurs à 1500 bytes ne représentent que 1,3% du volume total et les flux inférieurs à 10^6 bytes ne représentent que 34 % du volume total. **La plus grosse partie du volume total est donc constituée d'un petit nombre de "gros" flux.**

Les "gros" flux étant relativement peu nombreux et constituant l'essentiel du trafic, l'application des méthodes de hachage sur la trace utilisée risque de produire une très mauvaise répartition du trafic sur les différentes lignes de sortie. Il nous apparaît donc utile d'obtenir plus d'informations sur ces "gros" flux, notamment concernant l'évolution de leur débit et de leur nombre au cours du temps. Nous représentons, à la figure 4.8, les débits instantanés, sur la ligne d'*entrée* du routeur, des flux de longueur ≥ à 1500 bytes et des flux de longueur ≥ à 10^6 bytes. Nous indiquons également l'évolution du débit de l'ensemble des flux et du débit des flux < à 10^6 bytes sur la même figure.

Les flux de longueur supérieure ou égale à 1500 bytes (+/- 20% du nombre total des flux et 99,7% du volume total) approximent remarquablement bien l'ensemble des flux. Les "gros" flux de longueur supérieure ou égale à 10^6 bytes (+/- 0,5 % du nombre total des flux et 66% du volume) n'approximent pas bien l'ensemble des flux mais reflètent néanmoins très bien l'évolution du trafic correspondant à l'ensemble des flux. Le rapport entre les "gros" flux et l'ensemble des flux est relativement constant au cours du temps, le trafic correspondant aux flux inférieurs à 10^6 bytes étant relativement constant (figure 4.8).

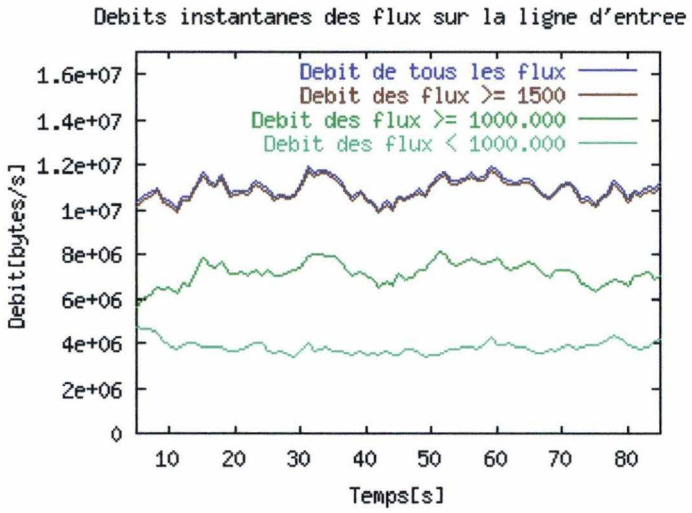


Figure 4.8 :
Débits instantanés de tous les flux, des flux ≥ 1500 bytes, des flux ≥ 10^6 bytes et des flux < 10^6 bytes sur la ligne d'entrée du routeur.

Il est donc possible de connaître relativement bien l'évolution du trafic total, uniquement grâce aux "gros" flux.

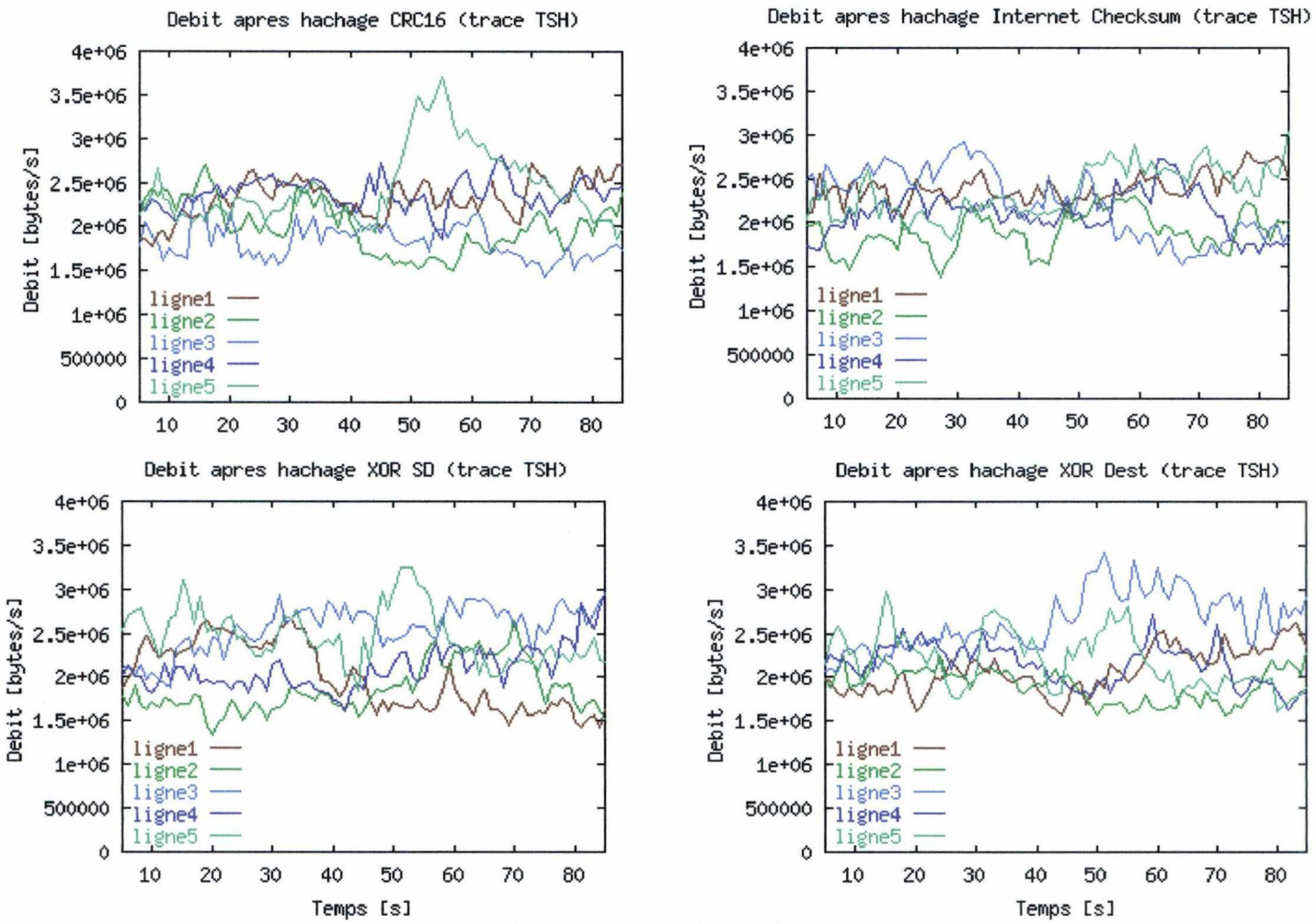
4.2 Simulation sur la trace TSH

Nous simulons les "bonnes" méthodes de répartition de trafic déterminées au chapitre 2, c'est-à-dire les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR IP SOURCE-DESTINATION, XOR IP DESTINATION et IP DESTINATION. Ces méthodes de hachage fonctionnent bien quand le trafic est uniformément distribué (cfr. chapitre 2). Nous allons examiner si elles fonctionnent aussi bien quand le trafic n'est pas uniformément distribué, ce qui est le cas de la trace TSH utilisée.

4.2.1 Répartition du trafic sur les différentes lignes de sortie au cours du temps

4.2.1.1 Etude qualitative

Les figures 4. 9 à 4.12 montrent la répartition de trafic obtenue après application des méthodes de hachage CRC16, INTERNET CHECKSUM, XOR IP SOURCE-DESTINATION, ET XOR IP DESTINATION.



Figures 4 - 9 à 4-12 :
Débit instantané de la charge de trafic après application des algorithmes de hachage CRC16, INTERNET CHECKSUM, XOR IP SOURCE-DESTINATION et XOR IP DESTINATION.

Nous constatons des répartitions *globales* de trafic relativement proches pour les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE-DEST et XOR DESTINATION : sur toute la durée de la mesure, les charges globales de trafic envoyées sur les différentes lignes de sortie semblent relativement proches. Il n'en est pas de même pour la méthode IP destination (figure 4.14). La répartition *globale* du trafic semble donc acceptable, excepté pour la méthode IP DESTINATION. La méthode INTERNET CHECKSUM semble, sur l'intervalle de temps de la mesure (+/- 90 s), un peu mieux répartir le trafic que les autres. *Il est difficile de départager les méthodes de hachage* CRC16, XOR SOURCE DESTINATION *et* XOR DESTINATION *sur base des graphiques obtenus.*

Nous constatons des répartitions *instantanées* de trafic peu uniformes. Nous observons par exemple à la figure 4-9 (CRC16) au temps t = 57 s, des débits instantanés de trafic très différents sur les différentes lignes de sortie, allant de 1,6 10^6 à 3,7 10^6 Bytes/s. Les plus petits écarts instantanés entre les lignes de sortie sont observés pour la méthode de hachage INTERNET CHECKSUM. Les méthodes de hachage CRC16, XOR IP SOURCE-DEST et XOR IP DESTINATION ont leurs plus gros écarts instantanés au cours de la même période (entre 45 et 65 s). Nous soupçonnons un ou plusieurs "gros" flux d'en être la cause. Nous le vérifions pour la méthode CRC16 qui présente un "gros" pic durant l'intervalle de temps [45, 65] secondes sur sa ligne de sortie numéro 5 (figure 4.9). La figure 4.13 nous montre que quatre "gros" flux seulement sont responsables de l'essentiel du "gros" pic. Ces 4 flux représentent ensemble 7,5 % du trafic total de la ligne de sortie n°5. Le flux apportant la plus grande contribution est également indiqué sur cette figure. Ce flux représente +/- 2 % du trafic total de la ligne de sortie n°5.

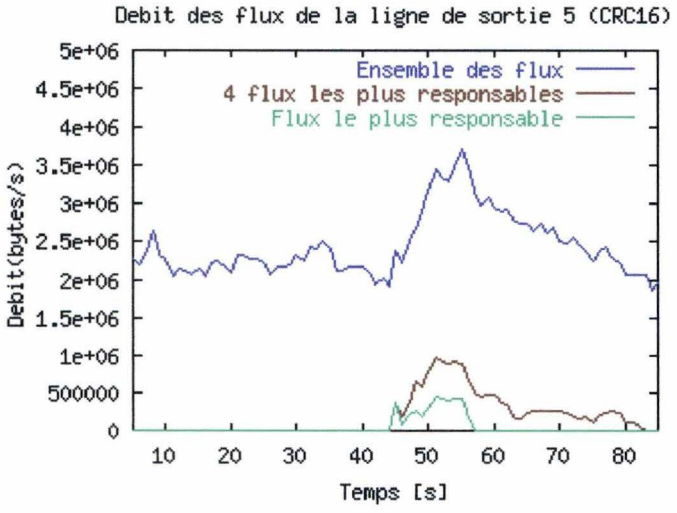


Figure 4.13 :
Débit des flux responsables du "gros pic" de la ligne 5 après hachage CRC16

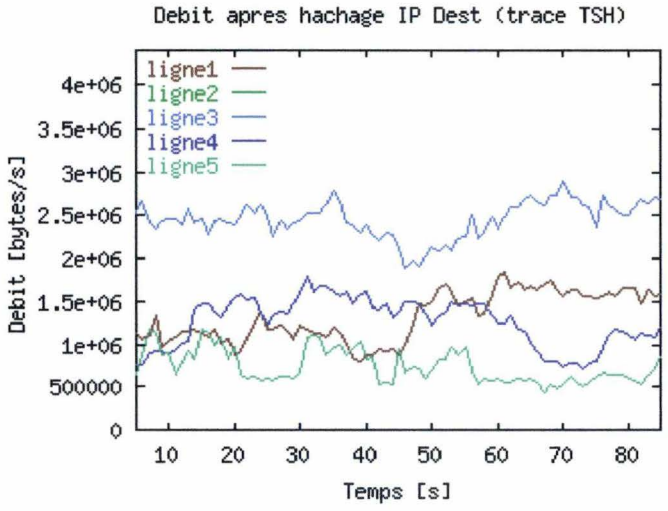


Figure 4.14 :
Répartition du trafic instantané après hachage IP DESTINATION

Les très "mauvaises" répartitions globales et instantanées du trafic après application de la méthode de hachage IP DESTINATION s'expliquent si on se souvient de la très mauvaise répartition des adresses IP destination anonymisées observée à la section 4.1.3. La méthode

de hachage IP DESTINATION tient en effet *uniquement* compte de l'adresse IP DESTINATION de chaque paquet IP lors de l'attribution du numéro de la ligne de sortie.

Les méthodes XOR IP DESTINATION et XOR IP SOURCE DESTINATION donnent de meilleurs résultats que la méthode IP DESTINATION parce qu'elles "hachent" l'adresse de destination en plusieurs morceaux et appliquent ensuite l'opérateur logique XOR sur les différents morceaux.

Les méthodes CRC16 et INTERNET CHECKSUM devaient en principe fournir les meilleures répartitions de trafic, le calcul des numéros de lignes de sortie faisant intervenir un nombre de bits beaucoup plus élevé (104 bits). La méthode INTERNET CHECKSUM répartit effectivement le trafic un peu mieux que les autres. Cela aurait également été le cas pour CRC16 s'il n'y avait pas eu le "pic" de la ligne 5 durant l'intervalle de temps [45, 65] secondes. Le "pic" de la ligne 5 étant essentiellement dû à un seul flux, nous ne pouvons pas le considérer comme déterminant pour départager les méthodes CRC16 et INTERNET CHECKSUM.

4.2.1.2 Etude quantitative

*a. Répartition **globale** du trafic sur chaque ligne de sortie*

Nous calculons tout d'abord le *pourcentage de trafic* obtenu pour chaque ligne de sortie pendant toute la durée de la mesure, c-à-d durant la période de +/- 90 s. Nous calculons ensuite *l'écart-type* des pourcentages de trafic obtenus. Nous obtenons le tableau 4 - 1 (qui porte donc sur toute la durée de la mesure).

	Pourcentage de trafic (%)					*Ecart-type*
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*	
CRC 16	21	18,7	16,6	21,6	22,1	2,1
INTERNET CHECKSUM	21,9	17,3	20,4	19	21,4	1,7
XOR SOURCE - DEST	18,1	16,9	23,3	19,4	22,3	2,4
XOR DESTINATION	19,1	17,7	24,0	19,5	19,7	2,1
DESTINATION	12,0	47,4	22,7	11,3	6,6	14,8

Tableau 4 - 1 :
Pourcentage de trafic obtenu pour chaque ligne de sortie pendant toute la durée de la mesure et écart-type des pourcentages de trafic obtenus.

Nous obtenons des pourcentages de trafic relativement équilibrés et des écart-types relativement faibles, pour toute la durée de la mesure, pour les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE-DEST et XOR DESTINATION. Seule la méthode de hachage IP DESTINATION produit un très mauvais balancement du trafic. Nous avons déjà expliqué pourquoi dans l'analyse qualitative.

De manière générale, la faible différence obtenue entre les écart-types ne permet pas de départager les méthodes CRC16, XOR SOURCE-DEST, XOR IP DESTINATION et XOR

DESTINATION. *Il faudrait effectuer des mesures sur des traces réelles (non anonymisées) et sur des intervalles de temps plus longs.*

*b. Répartition **instantanée** du trafic sur chacune des lignes de sortie*

Le trafic envoyé sur chaque ligne de sortie est-il plus ou moins constant ou présente-t-il de grandes variations au cours du temps ?

Nous essayons de répondre à la question en calculant **l'écart moyen relatif** (tableau 4-2) de chaque ligne de sortie. Nous définissons l'*écart moyen* comme étant égal à la *moyenne des valeurs absolues des écarts instantanés entre le débit instantané du trafic et son débit moyen. L'écart moyen relatif* est défini comme étant l'*écart moyen* divisé par le *débit moyen* sur l'ensemble de la mesure. L'*écart moyen relatif* mesure donc, pour une ligne donnée, l'uniformité, en moyenne, du trafic au cours du temps : plus *l'écart moyen relatif* est faible, plus le débit instantané a, en moyenne, des valeurs proches du débit moyen.

Nous calculons l'*écart moyen relatif* pour chaque ligne de sortie et pour chaque méthode de hachage au tableau 4-2. Nous calculons également la *moyenne des écarts relatifs* pour chaque méthode de hachage.

	Ecart moyen relatif (%)					*Moyenne des écarts relatifs*
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*	
CRC 16	8,4	12,5	8,8	6	13	9,8
Internet Checksum	6,0	9,3	14,7	9,9	10,7	10,1
XOR Source - Dest	16,7	11,5	9,5	9,8	10,3	11,6
XOR Destination	10,2	8,8	10,5	9,2	13,2	10,4
Destination	20,1	4,4	7,0	20,7	23,6	15,2

Tableau 4 - 2 :

Ecart moyen relatif pour chaque ligne de sortie et moyenne des écarts relatifs pour chaque méthode de hachage.

Les *moyennes* des écarts moyens relatifs sont fort semblables pour les méthodes CRC16, INTERNET CHECKSUM, XOR IP SOURCE-DEST, ET XOR IP DESTINATION. Elles semblent montrer une répartition relativement uniforme du trafic instantané sur l'ensemble des lignes de sortie. Les meilleurs résultats sont obtenus pour les méthodes CRC16 et INTERNET CHECKSUM. La méthode de répartition XOR DESTINATION est de nouveau "à la traîne" par rapport aux autres. La moyenne des écarts relatifs est de 50% plus élevée.

La comparaison de *l'écart maximum* entre les débits instantanés des différentes lignes de sortie est montrée au tableau 4-3. Le meilleur résultat est obtenu avec la méthode INTERNET CHECKSUM.

	CRC16	INTERNET CHECKSUM	XOR SOURCE DEST	XOR DEST	DEST
Ecart maximum entre les débits instantanés (bytes/s)	$2{,}1\ 10^6$	$1{,}3\ 10^6$	$1{,}6\ 10^6$	$1{,}7\ 10^6$	$2{,}2\ 10^6$

Tableau 4-3 :
Ecart maximum entre les débits instantanés des différentes lignes de sortie.

Des résultats semblables *globalement* ne veulent cependant pas dire que la répartition instantanée du trafic est acceptable pour chaque ligne de sortie *considérée en particulier.*
Nous observons par exemple, au tableau 4-2, un écart moyen relatif relativement important (14,7%) en ce qui concerne la ligne n°3 après application de la méthode de hachage INTERNET CHECKSUM. Les écarts moyens relatifs sont, de plus, fort différents d'une ligne de sortie à l'autre. Il suffit de calculer la *dispersion* (max-min) des écarts moyens relatifs pour chaque ligne de sortie pour s'en convaincre (tableau 4-4). Nous calculons également la *dispersion relative* (dispersion divisée par la moyenne des écarts relatifs).

	CRC16	INTERNET CHECKSUM	XOR SOURCE DESTINATION	XOR DEST	IP DEST
Dispersion	7	8,7	7,2	4,4	19,2
Dispersion relative (%)	71 %	86 %	62 %	42 %	126 %

Tableau 4-4 :
Dispersion (max-min) des écarts moyens relatifs.

Nous obtenons de très "mauvais" résultats pour toutes les méthodes de hachage La palme revient, une fois de plus, à la méthode IP DESTINATION.

Le trafic instantané envoyé sur une ligne de sortie déterminée peut donc présenter de fortes variations au cours du temps. Ces variations de trafic instantané peuvent être très différentes d'une ligne de sortie à l'autre.

c. Interprétation

Nous avons montré à la section 4.1.3 que la distribution du volume de trafic en fonction des adresses IP et que la distribution du nombre de flux en fonction des adresses IP étaient fort peu uniformes. Nous avons montré à la section 4.1.4 que la distribution des flux en fonction de leur longueur était également peu uniforme. Nous avons donc utilisé de "mauvaises" traces du point de vue de l'homogénéité.
Nous obtenons malgré tout de *"bons" résultats* en ce qui concerne *la répartition globale* du trafic *sur les différentes lignes de sortie*, du moins en ce qui concerne les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE-DEST et XOR DESTINATION. Les résultats obtenus ne permettent cependant pas de départager avec certitude ces méthodes de hachage. Les résultats obtenus pourraient être différents pour d'autres traces de trafic. *Il serait intéressant de pouvoir effectuer des mesures sur des temps plus longs* et sur des traces *réelles* (non anonymisées).

Les résultats obtenus sont *nettement moins "bons"* en ce qui concerne *la répartition instantanée* du trafic sur chacune des lignes de sortie. Ils peuvent s'expliquer si on se souvient que le trafic issu de la trace TSH est essentiellement *dû à quelques "gros" flux* (les flux $\geq$ à 10^6 bytes représentent 66 % du volume total) *peu nombreux* (les flux $\geq 10^6$ bytes ne représentent que 0,5 % du nombre total des flux). Les "gros" flux sont donc déterminants. Un seul "gros" flux peut même être déterminant comme l'a montré la figure 4.13. Il n'est donc pas étonnant d'observer une répartition instantanée du trafic fort inhomogène sur les différentes lignes de sortie.

Si nous étions un fournisseur d'accès ou un constructeur de routeur devant implémenter un mécanisme de LOAD BALANCING, *nous préférerions utiliser une technique simple, moins performante mais plus prévisible comme* XOR DESTINATION *: il vaut mieux utiliser la technique la moins mauvaise dans le pire des cas que la meilleure en moyenne ...*

4.2.2 Occupation des buffers des différentes lignes de sortie

Une autre manière d'évaluer les performances des différentes méthodes de répartition du trafic est de s'intéresser à l'occupation des "buffers" des différentes lignes de sortie au cours du temps. Le nombre de paquets présents dans le buffer est lié au mécanisme de régulation du trafic utilisé dans le routeur. Nous supposons un mécanisme de régulation de type TOKEN BUCKET[PER99].

Dans cette étude, les différentes méthodes de hachage ne sont pas appliquées directement sur le trafic sortant d'un routeur, où il existe un buffer par ligne de sortie, mais bien sur une trace anonymisée. Nous n'avons pas de buffer réellement dédié à chaque ligne de sortie. Nous simulons la présence de ces buffers par un ensemble de TOKEN BUCKET, chacun correspondant à une ligne de sortie. La taille des buffers n'étant pas connue à priori, nous considérons chaque buffer comme ayant une taille tout juste suffisante pour accepter tous les paquets IP qui arrivent.

4.2.2.1 TOKEN BUCKET

Un TOKEN BUCKET peut être représenté comme étant un "sceau percé" alimenté par un "robinet" amenant des crédits à un débit constant r, où r est le débit moyen de la ligne de sortie (figure 4. 15).

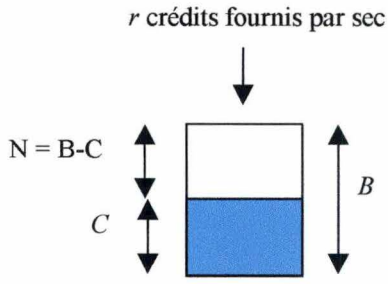


Figure 4.15 : TOKEN BUCKET

On ajoute des crédits à condition que le sceau ne soit pas plein : le nombre C de crédits accumulés dans le sceau est égal *au maximum* à B, où B est la taille du TOKEN BUCKET.
Lorsqu'un paquet IP arrive, l'algorithme vérifie si la taille de ce dernier est inférieure ou égale au nombre de crédits se trouvant dans le sceau. Si c'est le cas, le paquet est envoyé sur la ligne de sortie et le nombre de TOKEN présents dans le sceau est décrémenté de la taille du paquet IP. Sinon, le paquet est jeté ou stocké dans le buffer :

a) le paquet est jeté dans le cas où le buffer est plein
b) sinon, le paquet est stocké dans le buffer jusqu'à ce que le sceau ait accumulé suffisamment de crédits pour pouvoir l'envoyer sur la ligne de sortie.

Un TOKEN BUCKET permet ainsi de contrôler le volume de trafic sur une ligne de sortie ainsi que le débit avec lequel il est transmis. Il borne également la taille du buffer utilisé.

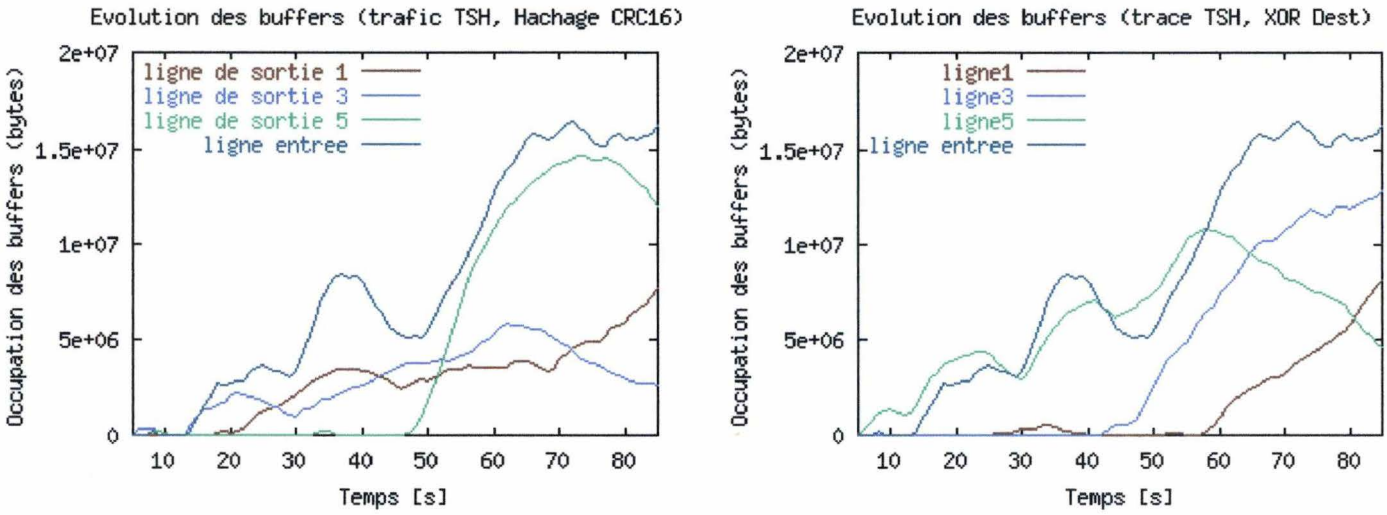
4.2.2.2 Utilisation du TOKEN TUCKET dans le cadre de notre étude

Dans cette étude, nous simulons les différentes méthodes de hachage sur une trace anonymisée (trace TSH). Les "lignes de sortie" sur lesquelles le trafic est réparti après application des méthodes de hachage ne sont donc pas des lignes réelles. La taille des buffers associés aux différentes lignes de sortie n'est pas connue à priori. Nous considérons, dans cette étude, chaque buffer comme ayant une taille tout juste suffisante pour pouvoir accepter tous les paquets IP qui lui arrivent. Chaque buffer est simulé par un TOKEN BUCKET. Il y a donc un TOKEN BUCKET par ligne de sortie. La taille B de chaque TOKEN BUCKET dépend de la ligne de sortie à laquelle il est associé. Elle est égale à la taille *minimale* que doit avoir le TOKEN BUCKET pour que le nombre de crédits présents dans le sceau soit toujours suffisant pour accepter chaque paquet IP qui arrive. Au cours de la lecture du fichier TSH, le nombre de crédits C accumulés dans le TOKEN BUCKET varie de 0 à B.

Pour chaque ligne de sortie, nous considérons le débit moyen r, calculé à partir de la trace TSH, comme étant le débit auquel nous voulons envoyer le trafic sur la ligne de sortie. Le "robinet" du TOKEN BUCKET amène donc les crédits au débit moyen r de la ligne de sortie auquel il est associé. Il arrive que le nombre de crédits amenés par le "robinet" ne soit pas suffisant pour compenser le nombre de crédits nécessaires à l'envoi des paquets sur la ligne de sortie. Ceci correspond à une situation où des paquets doivent être stockés dans le "buffer". La valeur $N = B - C$ correspond au nombre de crédits qu'il a fallu puiser dans le sceau pour stocker les paquets dans le "buffer". Le nombre de bytes correspondant aux paquets présents dans le buffer est donc égal à $N = B - C$ (figure 4.15). Une valeur de N égale à *zéro* correspond à un TOKEN BUCKET "plein" de crédits et donc à un buffer vide. Ceci signifie que le débit de la ligne de sortie est inférieur, depuis un certain temps, au débit moyen. Une valeur de N supérieure à *zéro* signifie que le nombre de crédits utilisés pour l'acceptation des paquets est supérieur, depuis un certain temps, au nombre de crédits fournis par le robinet au débit moyen de la ligne de sortie. Le débit instantané de la ligne est donc supérieur, depuis un certain temps, au débit moyen. Plus N augmente, plus on se rapproche d'une situation de congestion.

4.2.2.3 Résultats obtenus

A titre d'exemples, nous montrons l'évolution de l'occupation instantanée des buffers ($N = B - C$), au cours du temps, pour les lignes de sortie 1, 3 et 5 après application des méthodes CRC16 et XOR IP DESTINATION (figures 4.16 et 4.17). Nous montrons également l'évolution instantanée du buffer d'entrée sur les deux figures.



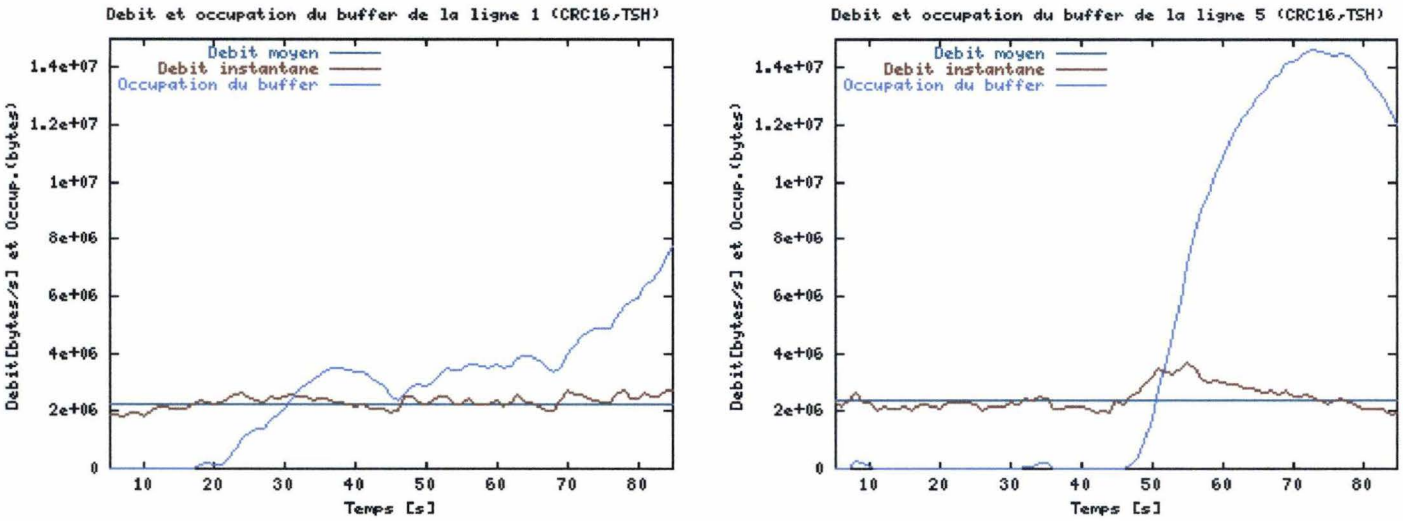
Figures 4.16 et 4.17 :
Occupation des buffers de sortie après application des hachages CRC16 et XOR DESTINATION

Les différents pics obtenus concernant l'occupation instantanée des buffers sont repris au tableau 4-5.

	Ligne d'entrée	*Lignes de sortie CRC16*			*Lignes de sortie XOR IP Destination*		
		Ligne 1	*Ligne 4*	*Ligne 5*	*Ligne 1*	*Ligne 4*	*Ligne 5*
Pics	+/- 1,6 10^7	+/- 7,8 10^6	+/- 4 10^6	+/- 1,4 10^7	+/- 1,2 10^7	+/- 6 10^6	+/- 1,2 10^7

Tableau 4 - 5 : Pics obtenus concernant l'occupation instantanée des buffers

L'occupation maximale des buffers après application des méthodes de hachage est du même ordre de grandeur que l'occupation maximale du buffer de la ligne d'entrée. Les graphiques obtenus ne sont pas étonnants si on les compare avec les figures de répartition instantanée du trafic obtenues à la section 4.2.1.1. A titre d'exemple, nous montrons, sur les mêmes graphiques, l'évolution du débit instantané et l'évolution de l'occupation instantanée des buffers pour les lignes de sortie 1 et 5 après application de la méthode de hachage CRC16 (figures 4.18 et 4.19).



Figures 4.18 et 4.19 :
Débit instantané et occupation instantanée du buffer des lignes 1 et 5 après hachage CRC16.

4.2.2.4 Interprétation des résultats obtenus

1) L'occupation du buffer de la ligne de sortie n°1 (figure 4.18) s'explique facilement :

a) Pour $0 < t < 18$ s : le débit instantané est inférieur au débit de la ligne de sortie. Aucun paquet n'a donc besoin d'être stocké. Nous constatons donc un buffer vide pour la ligne de sortie durant cet intervalle de temps
b) Pour $18 <= t < 39$ s : le débit instantané est supérieur au débit moyen. Les paquets vont donc s'accumuler dans le buffer. Il est donc normal d'observer une croissance continue du graphique d'occupation instantanée du buffer de la ligne de sortie durant cet intervalle de temps.
c) Pour $39 <= t < 47$ s : le débit instantané est inférieur au débit moyen. Le buffer va donc se vider peu à peu. Nous comprenons donc la décroissance continue de l'occupation instantanée du buffer de la ligne de sortie n°1 durant cet intervalle de temps.
d) Pour $47 <= t < 69$ s : le débit instantané est en moyenne légèrement supérieur au débit moyen durant cette période. Les paquets s'accumulent donc relativement lentement à l'intérieur du buffer.
e) Pour $69 <= t < 86$ s : le débit instantané est nettement supérieur au débit moyen. Les paquets s'accumulent donc à nouveau dans le buffer. Nous constatons donc une croissance continue du graphique d'occupation instantanée du buffer de la ligne de sortie durant cet intervalle de temps.

2)Le graphique d'occupation instantanée du buffer de la ligne de sortie n°5 s'explique de la même manière. Nous constatons cependant une amplitude beaucoup plus grande qu'à la figure 4.18. Ceci résulte des écarts en moyenne beaucoup plus importants entre le débit instantané et le débit moyen durant l'intervalle de temps [47, 74] sec.

3)Les fortes variations d'occupation instantanée des buffers observées découlent des "mauvaises" répartitions de trafic observées à la section 4.2.1. L'essentiel du trafic issu de la trace TSH est essentiellement dû à quelques "gros[36]" flux. Ces "gros" flux étant peu nombreux, leur répartition est inhomogène.

4) *L'occupation des buffers variant fortement au cours du temps et d'une ligne de sortie à l'autre, il semble difficile d'évaluer les performances des différentes méthodes de répartition de trafic sur base de cet unique critère.* Il y a tout lieu de penser que si la trace avait été captée durant les 90 secondes suivantes (par exemple), nous aurions eu des graphiques d'occupation des buffers totalement différents.

5) Le fait que l'occupation maximale des buffers des différentes lignes de sortie soit du même ordre de grandeur que l'occupation maximale du buffer de la ligne d'entrée nous amène à penser que *le* LOAD BALANCING *n'est pas en elle-même une technique très efficace pour minimiser l'occupation des buffers.* Ceci n'est bien entendu pas crucial puisque ce n'est pas l'objectif principal[37] du LOAD BALANCING. Les "mauvais" chiffres concernant l'occupation maximale des buffers sur les différentes lignes de sortie peuvent également s'expliquer à partir du petit nombre de " gros" flux :

a) *tous les paquets issus d'un même flux doivent obligatoirement se diriger vers la même ligne de sortie du routeur.* Sinon, nous risquerions de fausser les mécanismes de contrôle de congestion, ce qui obligerait le mécanisme TCP à retransmettre inutilement des paquets, provoquant une augmentation du trafic et donc une dégradation inutile des débits réels. Ceci ne pose aucun problème en ce qui concerne les "petits" flux, la plupart de ceux-ci n'étant constitués que d'un seul paquet (cfr. section 4.1.4).

b) cette situation provoque, pour les très gros flux, une forte augmentation du débit instantané de la ligne de sortie vers laquelle ils sont dirigés, provoquant ainsi une forte accumulation de paquets dans le buffer de cette ligne de sortie. Le risque de congestion du buffer existe et des paquets risquent éventuellement d'être supprimés.

c) Le nombre de gros flux étant peu élevé (ils ne correspondent qu'à 0,5 % du total des flux), le déséquilibre entre l'occupation des buffers des différentes lignes est relativement important.

6) Le fait que l'occupation maximale des buffers des différentes lignes de sortie soit du même ordre de grandeur que l'occupation maximale du buffer de la ligne d'entrée n'est pas un problème propre aux méthodes de LOAD BALANCING. Le routage IP "classique" (quand un paquet arrive, le routeur regarde le contenu du paquet et choisit une ligne de sortie en fonction de l'adresse de destination du paquet et du contenu de la table de routage) connaît le même problème : le nombre de gros flux étant peu élevé, le déséquilibre entre l'occupation des buffers des différentes lignes est relativement important.

---

[36] De tailles comprises entre 10^6 et 10^7 bytes.

[37] L'objectif principal du LOAD BALANCING est de répartir le mieux possible le trafic entre plusieurs chemins de coût minimum de manière à diminuer au maximum le nombre de points de congestion dans le réseau.

Le problème lié à la taille des buffers des différentes lignes de sortie n'est donc pas *provoqué* par les méthodes de LOAD BALANCING. Ces méthodes *subissent* le problème. Elles ne parviennent malheureusement pas à le résoudre.

4.3 Simulation sur la trace NETFLOW

L'intérêt des résumés de flux NETFLOW réside dans le fait qu'ils permettent d'évaluer les différentes méthodes de répartition de trafic sur des intervalles de temps relativement grands, de l'ordre de plusieurs heures ou plusieurs jours (cfr. chapitre 3). Ces résumés de flux NETFLOW se font bien entendu au détriment de la précision et donc de la finesse de l'analyse.

4.3.1 Transformation de la trace TSH en une trace NETFLOW

Nous n'avons pas réussi à obtenir une trace NETFLOW de la part d'un gestionnaire de réseau. Nous avons *simulé* NETFLOW à partir de la trace TSH. Un script a été réalisé pour créer, à partir de la trace TSH, une trace possédant des informations de même type que NETFLOW, c'est-à-dire un résumé des informations pour chaque flux et pour toute la période de mesure (90,03 s).

4.3.2 Simulation du trafic à partir de la trace NETFLOW obtenue

Pour simuler le trafic à partir de la trace NETFLOW, nous avons considéré chaque flux de la trace NETFLOW comme étant un "*fluide parfait*" débitant de manière continue une *certaine quantité de trafic* chaque seconde de son existence. Plusieurs flux peuvent évidemment diffuser simultanément sur la même ligne de sortie. On pourrait critiquer cette approximation en disant qu'un *vrai* routeur ne procède pas comme cela. N'oublions cependant pas qu'il s'agit d'une simulation et que nous travaillons de toute façon ici avec des *résumés de flux*, pas avec du trafic réel.

La *quantité de trafic* émise chaque seconde (dixième, centième ou millième de seconde) par chaque "fluide parfait" est égale au nombre total de bytes appartenant au flux correspondant divisé par le temps d'existence (durée) du "*fluide parfait*".

Le *temps d'existence* de chaque "*fluide parfait*" est exprimé en secondes (dixième, centième ou millième de seconde). Il est calculé à partir des TIMESTAMP de début et de fin du flux correspondant de la manière suivante:

Temps d'existence = *valeur entière du* TIMESTAMP *de fin du flux*
- *valeur entière du* TIMESTAMP *de début du flux*
+ *1*

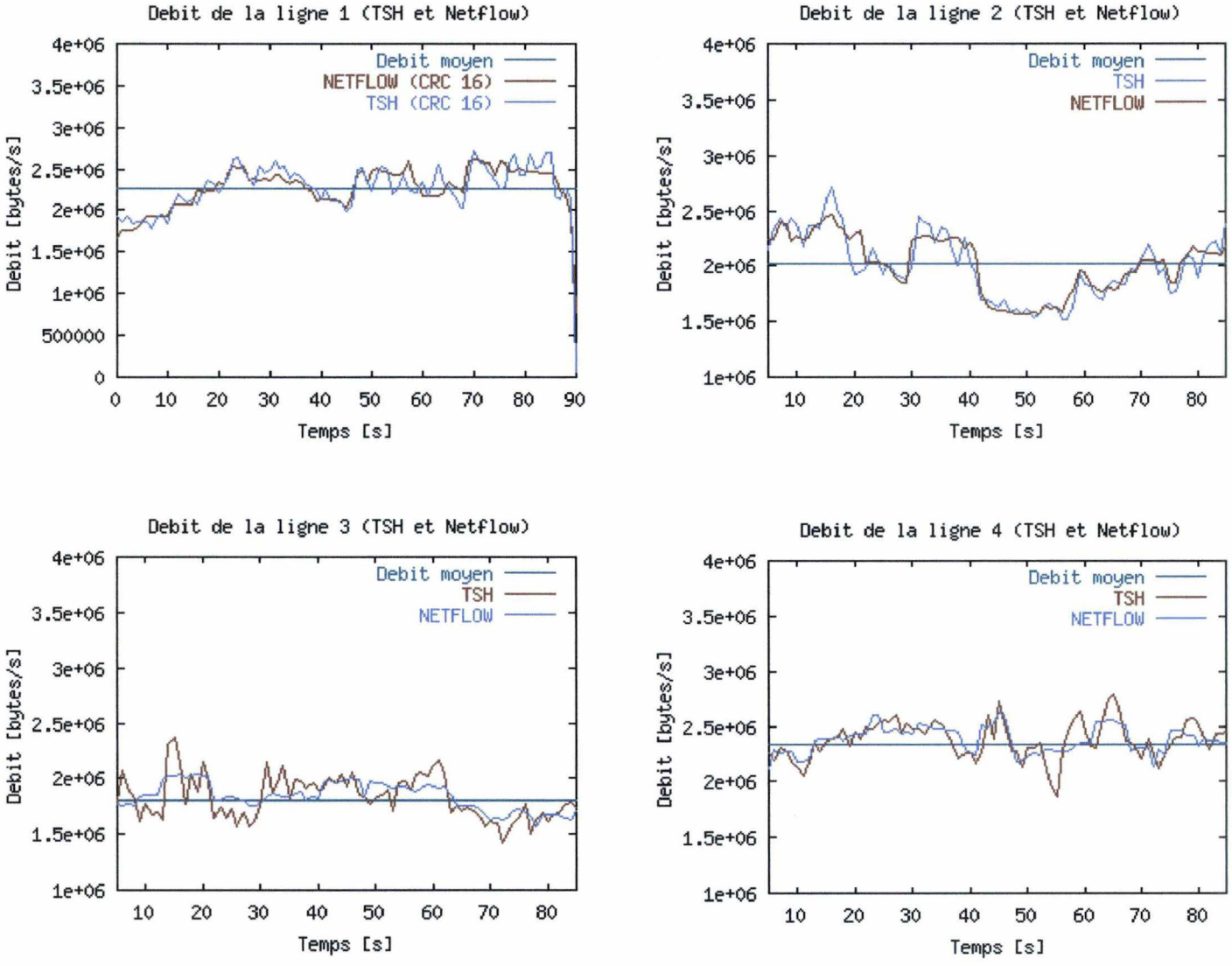
Le temps de démarrage de chaque "*fluide parfait*" est égal à la valeur entière du temps de démarrage du flux correspondant.

4.3.3 Résultats obtenus et comparaison des performances

4.3.3.1 Evolution du débit sur chaque ligne de sortie au cours du temps

a. Etude qualitative

Les *figures* 4.20 à 4.23 montrent, à titre d'exemple, l'évolution du débit NETFLOW sur les lignes de sortie 1, 2, 3 et 4 pour la méthode de hachage CRC16. Pour faciliter les comparaisons, l'évolution du débit TSH, pour la même ligne, est également représentée sur ces graphiques.



Figures 4.20 à 4.23 :
Comparaison des débits instantanés obtenus sur les lignes de sortie 1, 2, 3, et 4 après application du hachage CRC16 sur les traces TCPDUMP et NETFLOW

Nous constatons une forte ressemblance de l'évolution, au cours du temps, des trafics issus des traces TSH et NETFLOW sur chaque ligne de sortie.

La répartition *globale* de trafic observée avec NETFLOW semble très proche de celle observée avec TSH. En principe, la répartition globale devrait même être identique :

a) Chaque byte envoyé sur une ligne de sortie déterminée est repris dans un paquet (trace TSH) ou dans un flux (trace NETFLOW).

b) Chaque paquet (trace TSH) est repris dans un résumé de flux (trace NETFLOW) en fonction du flux auquel il appartient

c) Chaque byte est donc envoyé sur la même ligne de sortie, qu'il soit repris dans un paquet (trace TSH) ou dans un résumé de flux (trace NETFLOW)

d) Le nombre de bytes envoyés sur une ligne de sortie déterminée est donc identique pour les traces NETFLOW et TSH.

La répartition *instantanée* du trafic obtenue avec NETFLOW est différente de celle obtenue avec TSH. Ceci n'a rien d'étonnant : le trace de trafic NETFOW "lisse" en effet la trace de trafic TSH.

L'écart entre les deux répartitions instantanées ne paraît pas important. Nous essayons de quantifier de manière précise cette erreur dans la partie "Etude quantitative".

b. Etude quantitative

Nous calculons le *pourcentage de trafic* obtenu pour chaque ligne de sortie et pour chaque méthode de hachage. Nous obtenons, comme prévu à la section précédente, des *pourcentages* identiques à ceux obtenus après application des méthodes de répartition de trafic sur la trace TSH. *L'**écart moyen relatif*** (défini à la section 4.2.1.2) est calculé au tableau 4 - 6.

	Ecart moyen relatif (%)					*Moyenne des écarts relatifs(%)*
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*	
CRC 16	8,4	10,7	6,5	4,9	10,8	8,3
INTERNET CHECKSUM	4,1	7,8	13,9	8,7	9,3	8,8
XOR SOURCE-DEST	14,4	10,4	7,4	8,5	9,8	10,1
XOR DESTINATION	7,8	6,4	9,6	7,3	11,1	8,4
DESTINATION	20,3	3,9	5,2	18,9	22,4	14,1

Tableau 4.6 :
Ecart moyen relatif pour chaque ligne de sortie et moyenne des écarts relatifs pour chaque méthode de hachage

Les moyennes des écarts relatifs sont un peu plus faibles mais évoluent de la même manière que celles obtenues avec la trace TSH (cfr. tableau 4-2). Des résultats semblables *globalement* ne veulent cependant pas dire que les répartitions *instantanées* de trafic sont acceptables pour chaque ligne de sortie en particulier. Les écarts moyens relatifs peuvent être fort différents d'une ligne de sortie à l'autre. Nous calculons, toujours pour la trace NETFLOW, la ***dispersion*** (max-min) des écarts moyens relatifs de chaque ligne de sortie (tableau 4-7). Nous calculons

également la *dispersion relative.* Nous constatons qu'elle suit la même tendance que celle observée au tableau 4-4 pour le trafic TSH. NETFLOW *semble donc être une bonne approximation de* TCPDUMP *en ce qui concerne le* LOAD BALANCING.

	CRC16	INTERNET CHECKSUM	XOR SOURCE DESTINATION	XOR DEST	IP DEST
Dispersion	5,9	9,8	7	4,7	18,5
Dispersion relative (%)	71 %	111 %	69 %	56 %	131 %

Tableau 4-7 :
Dispersion (max-min) des écarts moyens relatifs de chaque ligne de sortie

Il nous paraît important de connaître de la manière la plus précise possible l'*erreur de répartition instantanée* commise en utilisant NETFLOW à la place de TCPDUMP. Ceci fait l'objet de la section suivante.

c. Mesure de l'erreur de répartition instantanée

Nous essayons de quantifier de manière précise l'*erreur de répartition instantanée du trafic* commise en mesurant *l'erreur moyenne* de *répartition instantanée du trafic.* Le calcul de l'erreur *moyenne* nous semble intéressant parce qu'il nous donne une idée de l'ordre de grandeur de l'erreur effectuée durant toute la durée de la mesure. Le calcul de *l'erreur moyenne* ne suffit cependant pas. Il faut également mesurer *l'écart maximum relatif entre les écarts instantanés des traces TSH et NETFLOW pour les cinq lignes de sortie et pour les différentes méthodes de hachage.* La connaissance de cette écart maximum est intéressante car elle permet de se faire une idée du débit maximal susceptible de se produire sur chaque ligne de sortie. Nous calculons à cet effet *l'écart maximum relatif entre les deux graphiques.*

1. Calcul de l'erreur moyenne

Le calcul de l'erreur *moyenne* donne une idée de l'ordre de grandeur de l'erreur effectuée durant toute la durée de la mesure. Nous déterminons cette *erreur moyenne* en calculant la *moyenne relative*, par rapport au débit moyen du trafic de la ligne de sortie, *de la valeur absolue des écarts instantanés* entre le trafic TSH et le trafic NETFLOW. Nous calculons également *l'écart de surface relatif* entre les deux graphiques.

a) Calcul de la **moyenne relative**, par rapport au débit moyen du trafic de la ligne de sortie, des valeurs absolues **des écarts instantanés** entre le trafic TSH et le trafic NETFLOW.

Le calcul de la *moyenne des valeurs absolues des écarts instantanés* nous semble intéressant parce qu'il nous donne une idée de l'erreur moyenne de répartition instantanée commise entre TSH et NETFLOW durant toute la durée de la mesure. Nous

exprimons les résultats au tableau 4-8 sous forme de moyenne *relative* de manière à relativiser l'erreur par rapport au débit moyen des différentes lignes de sortie.

	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC-16	5 %	5 %	6,5 %	4,5 %	6 %
Internet Checksum	4 %	7 %	5 %	5,5 %	5,5 %
XOR IP Source-Destination	6 %	6 %	4,5 %	5,5 %	6 %
XOR IP Destination	5 %	5 %	4,5 %	4 %	7 %
IP Destination	6,5 %	3 %	4 %	8 %	11 %

Tableau 4-8 :
Moyenne relative, pour chaque ligne de sortie, des valeurs absolues des écarts instantanés entre le trafic TSH et le trafic NETFLOW.

La moyenne relative est de 5,6 % si on tient compte de tous les résultats repris dans le tableau 4-8. La dispersion des moyennes relatives n'est donc pas exagérée.

b) Calcul de **l'écart de surface relatif** entre les deux graphiques.

L'écart de surface relatif est égal à l'aire comprise entre les deux graphiques divisée par la surface située sous le graphique TSH. Cette méthode correspond bien intuitivement à la manière dont nous procédons lorsque nous analysons les graphiques de manière qualitative. Nous ne pouvons malheureusement pas calculer *exactement* cet écart. Nous ne disposons en effet pas de données continues mais bien de données discrètes (le nombre de bytes reçus chaque seconde). Nous approximons donc l'aire comprise entre les deux graphiques à l'aide d'une série de rectangles issus de ces valeurs discrètes. Les résultats obtenus figurent au tableau 4-9.

	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC-16	5 %	5%	6,5 %	4,5 %	6 %
Internet Checksum	4,5 %	7 %	5,5 %	5 %	6 %
XOR IP Source-Destination	6 %	6 %	5 %	6 %	6 %
XOR IP Destination	5,5 %	5 %	5 %	4 %	7 %
IP Destination	6,5 %	3 %	4,5 %	8 %	11 %

*Tableau 4-9 : **Ecart de surface relatif** entre les deux graphiques de répartition instantanée*

L'écart de surface relatif est de 5,7 % si on tient compte de tous les résultats repris dans le tableau 4-9. La dispersion des moyennes relatives n'est donc pas exagérée.

La comparaison des tableaux 4-8 et 4-9 montre que le calcul de la *moyenne relative des écarts instantanés* et le calcul de *l'écart de surface relatif* aboutissent à des résultats fort proches. L'*erreur moyenne de répartition instantanée* semble pouvoir être obtenue aussi bien par le calcul de la *moyenne relative des écarts instantanés* que par le calcul de *l'écart de surface relatif.* NETFLOW semble être une bonne approximation de TCPDUMP en ce qui concerne la répartition du trafic sur les différentes lignes de sortie après application des méthodes de LOAD BALANCING. Nous pouvons donc envisager l'utilisation de NETFLOW à la place de TCPDUMP *pour mesurer la répartition de trafic* sur les différentes lignes de sortie durant des intervalles de temps relativement longs.

2. *Calcul de l'écart maximum relatif entre les écarts instantanés des traces TSH et NETFLOW pour les cinq lignes de sortie et pour les différentes méthodes de hachage (tableau 4-10).*

La connaissance de cette erreur est intéressante car elle permet, à partir de la mesure du débit maximal instantané obtenu avec NETFLOW, de se faire une idée du débit maximal réellement susceptible de se produire sur chaque ligne de sortie (tableau 4-10). La connaissance de cette erreur permet de borner le débit maximal susceptible de se produire sur chaque ligne de sortie.

	Ecart maximum relatif (%)				
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC 16	16	18,5	20	17,5	32
INTERNET CHECKSUM	17,5	24,5	20,5	22	22
XOR SOURCE DEST	17	19,5	20,5	20,5	24,5
XOR DEST	18,5	14	13	15,5	28,5
DEST	28,5	11,5	15,5	27	44

Tableau 4-10 : Ecart maximum relatif entre les écarts instantanés des traces TSH et NETFLOW pour les cinq lignes de sortie et pour les différentes méthodes de hachage.

L'erreur dans le *"pire des cas instantanés"* vaut en moyenne +/- 21 %. Si nous excluons la plus "mauvaise" méthode (cfr. section 4.2.1), c'est-à-dire la méthode IP DESTINATION, nous obtenons une erreur *moyenne* des "pires cas instantanés" de +/- 20% et une erreur *maximum* des "pires cas instantanés" de +/- 32 %.

4.3.3.2 Evolution du buffer de chaque ligne de sortie au cours du temps

a. Etude qualitative

A titre d'exemple, observons la figure 4.24 obtenue sur les lignes de sortie 1, 3 et 5 après application de la méthode de hachage CRC16. Les graphiques obtenus ont la même allure que ceux obtenus pour le fichier TSH (figure 4.16).

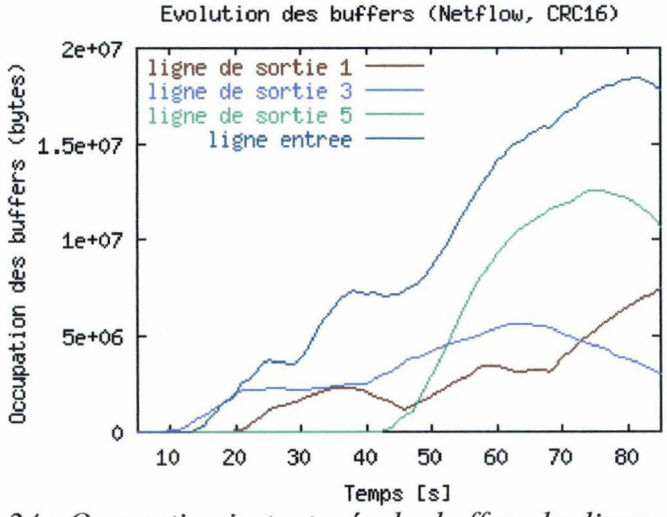
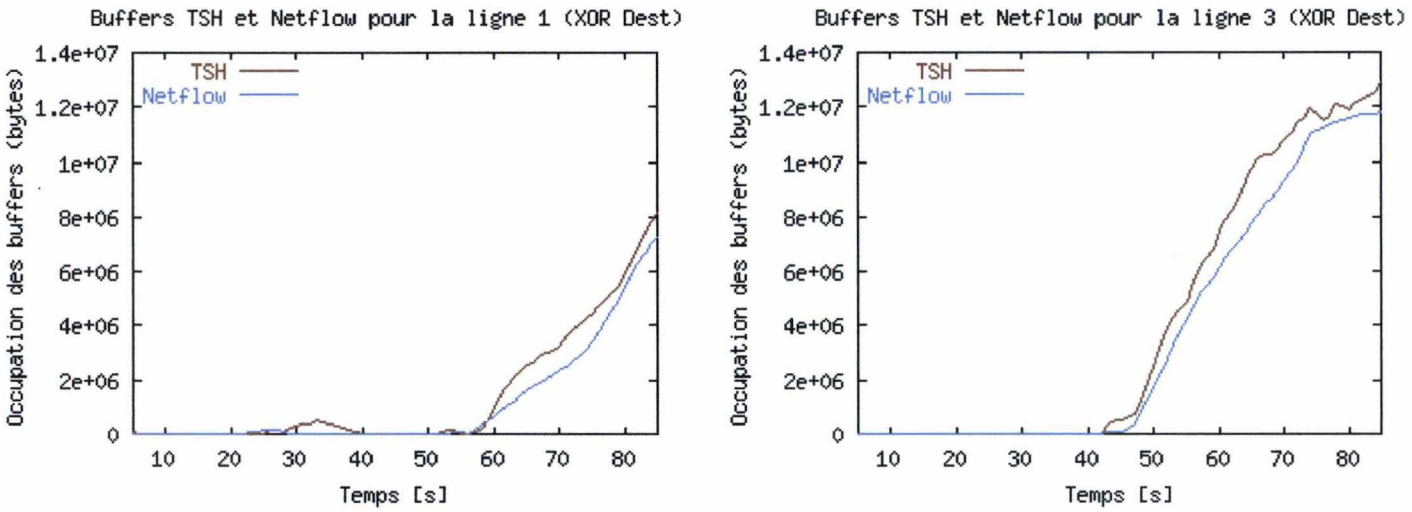


Figure 4.24 : Occupation instantanée des buffers des lignes de sortie 1, 3 et 5

Pour faciliter les comparaisons, on peut représenter sur le même graphique l'évolution des buffers TSH et NETFLOW pour chaque ligne de sortie. Les figures 4.25 et 4.26 comparent les graphiques obtenus pour les buffers des lignes de sortie 1 et 3 dans le cas particulier du hachage XOR IP DESTINATION.



Figures 4.25 et 4.26 :

Comparaison des occupations instantanées des buffers des lignes de sortie 1 et 3 après application du hachage XOR DESTINATION sur les traces TCPDUMP et NETFLOW

Nous constatons une ressemblance significative de l'évolution, au cours du temps, des buffers pour TSH et NETFLOW sur chaque ligne de sortie. L'évolution *instantanée* des buffers avec NETFLOW est "un peu" différente de celle obtenue avec TSH. Ceci n'a rien d'étonnant : NETFOW "lisse" en effet l'accumulation des paquets dans le buffer. L'écart relatif entre les deux occupations instantanées des buffers semble nettement *plus important* que celui obtenu entre les débits instantanés TSH et NETFLOW . Il nous paraît important de connaître de la manière la plus précise possible l'*erreur d'occupation instantanée* des buffers commise en utilisant NETFLOW à la place de TCPDUMP. Ceci fait l'objet de la section suivante.

b. Etude quantitative

Nous essayons de quantifier de manière précise l'erreur d'occupation instantanée des buffers effectuée en mesurant *l'erreur moyenne d'occupation instantanée* des buffers. Le calcul de cette *moyenne* nous semble intéressant parce qu'il nous donne une idée de l'ordre de grandeur de l'erreur effectuée durant toute la durée de la mesure. Le calcul de l'erreur moyenne ne suffit cependant pas. Il faut également mesurer *l'erreur dans le "pire cas instantané"*. La connaissance de cette erreur est intéressante car elle nous fournit un renseignement sur l'occupation maximale des buffers et donc sur la taille maximum des buffers à utiliser.

1. Calcul de l'erreur moyenne d'occupation des buffers

Nous calculons à cet effet la *moyenne des écarts relatifs instantanés* par rapport au débit moyen *des écarts instantanés* et *l'écart de surface relatif* entre les deux graphiques.

a) Calcul de la **moyenne des écarts relatifs instantanés**

On calcule pour chaque seconde la valeur absolue de l'*écart* entre l'occupation instantanée du buffer TSH et l'occupation du buffer NETFLOW. On *divise* ensuite chaque écart *obtenu par l'occupation instantanée du buffer TSH.* La moyenne de ces écarts relatifs instantanés est alors calculée. Les résultats obtenus figurent au tableau 4-11.

	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC-16	17 %	38 %	25 %	21 %	27 %
Internet Checksum	338 %	132 %	10 %	12 %	16%
XOR IP Source-Destination	18 %	22 %	68 %	25 %	21 %
XOR IP Destination	75 %	70 %	13 %	19 %	15 %
IP Destination	9 %	20 %	49 %	9 %	12 %

Tableau 4 -11 : Moyenne des écarts relatifs instantanés

Nous obtenons des chiffres très élevés, ce qui ne semble pas bien correspondre aux observations qualitatives de la section précédente. Ces chiffres élevés peuvent s'expliquer si l'on remarque que le calcul de la *moyenne* prend en considération tous les écarts relatifs instantanés *de la même manière*, que les buffers soient à peu près vide ou qu'ils soient relativement bien remplis. Or, nous pouvons constater en observant les graphiques (par ex. celui montrant l'évolution des buffers de la ligne 3 pour XOR DESTINATION) que les écarts relatifs entre NETFLOW et TCPDUMP sont précisément les plus importants quand le buffer est à peu près vide. Ceci n'a rien d'étonnant puisque utiliser NETFLOW à la place de TSH revient à «lisser », à «étaler » le trafic réel.

A titre d'exemple, calculons l'écart relatif, pour la ligne de sortie 3 après application de la méthode de hachage XOR DESTINATION, au temps t = 45 s. A ce moment, les deux buffers sont à peu près vides. Nous obtenons un écart relatif de 80%. Calculons ensuite l'écart relatif au temps t = 78 s, c'est-à-dire à un moment où les deux buffers sont relativement bien remplis. Nous obtenons un écart relatif de 5,5 %. Le problème vient du fait que le calcul de la *moyenne* des écarts relatifs instantanés accorde la même pondération aux deux écarts relatifs. La situation au temps t = 78 s est pourtant beaucoup plus représentative de ce qui se passe globalement au niveau de la différence d'occupation des buffers TSH et NETFLOW.

Il vaut donc mieux éviter de quantifier l'erreur en partant des écarts *relatifs* instantanés.

Remarquons encore que la situation au temps t = 78 s correspond à une situation critique, où on se rapproche de l'occupation maximale des buffers. C'est surtout à ce moment qu'il est important de connaître l'erreur entre l'occupation TSH et NETFLOW. Si nous constatons

une erreur faible entre TSH et NETFLOW aux moments où l'on se rapproche de l'occupation maximale des buffers, nous pourrons en déduire que l'on peut, avec une trace NETFLOW, connaître avec quasi autant de précision qu'avec une trace TSH, la quantité de buffer utilisée, après application des méthodes de LOAD BALANCING.

b) Calcul de la **moyenne relative**, *par rapport au débit moyen*, **des écarts instantanés**

On calcule pour chaque seconde la valeur absolue de l'*écart* entre l'occupation du buffer TSH et l'occupation du buffer NETFLOW. On additionne tous les écarts instantanés obtenus. On divise ensuite la somme par le *débit moyen* du trafic de la ligne de sortie. En divisant par le débit moyen plutôt que par l'occupation du buffer NETFLOW, on devrait pouvoir résoudre, au moins partiellement le problème soulevé au point a) précédent. La moyenne relative des écarts instantanés est également calculée. Les résultats obtenus sont repris au tableau 4-12.

	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC-16	20 %	22 %	20 %	12 %	33 %
Internet Checksum	18 %	33 %	34 %	16 %	18 %
XOR IP Source-Destination	38 %	14 %	45 %	6 %	32 %
XOR IP Destination	17 %	30 %	19 %	22 %	23 %
IP Destination	12 %	13 %	14 %	34 %	46 %

Tableau 4 -12 : Moyenne relative, par rapport au débit moyen, des écarts instantanés

Les chiffres obtenus sont moins élevés. Ils semblent mieux correspondre aux figures 4.25 et 4.26. Nous ne pouvons cependant pas être certain qu'ils reflètent bien l'erreur moyenne *réelle* d'occupation instantanée des buffers commise en utilisant NETFLOW à la place de TCPDUMP.

Il est possible d'utiliser d'autres méthodes pour mesurer l'erreur d'occupation instantanée des buffers. Ces méthodes ne prouveront cependant rien de plus. Il semble difficile de trouver une méthode dont nous puissions être sûrs qu'elle reflète à 100 % l'erreur réellement effectuée en utilisant NETFLOW à la place de TSH.

Nous pensons néanmoins que le calcul de *l'écart de surface relatif* entre les deux graphiques (qui fait l'objet du paragraphe suivant) devrait relativement bien approximer l'erreur effectuée car elle correspond tout à fait bien intuitivement à la manière dont nous procédons lorsque nous analysons les graphiques de manière qualitative.

c) Calcul de **l'écart de surface relatif** entre les deux graphiques.

La connaissance de cet écart est intéressant car il nous fournit un renseignement sur l'occupation maximale des buffers et donc sur la taille maximale des buffers à utiliser. Le concept d'écart de surface relatif a déjà été expliqué à la section 4.3.3.1. Les résultats obtenus figurent au tableau 4-13.

	Ecart de surface relatif (%)				
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC16	15	12	13	13	16
INTERNET CHECKSUM	56	30	9	9	15
XOR IP SOURCE-DESTINATION	22	25	12	12	9
XOR IP DESTINATION	22	25	12	12	9
IP DESTINATION	5	13	27	7	9

Tableau 4 - 13: ***Ecart de surface relatif*** *entre les graphiques TCPDUMP et NETFLOW d'occupation instantanée des buffers.*

Les écarts observés reflètent nos observations qualitatives des graphiques d'évolution instantanée des buffers après application des méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE-DESTINATION, XOR DESTINATION et IP DESTINATION. L'*erreur moyenne* est relativement élevée. Elle peut s'expliquer si on se souvient des différences obtenues entre les *débits instantanés* de TSH et de NETFLOW (cfr. section 4.3.3.1). Ces différences n'étaient pas très importantes (5,6 % en moyenne avec un "pic" de 7 % si on exclut la méthode de hachage IP DESTINATION). Elles peuvent cependant provoquer une variation importante de l'occupation des buffers, comme expliqué à la section 4.2.2.4.

2. *Calcul de l'erreur dans le "pire cas instantané"*

Cette erreur nous fournit un renseignement sur l'occupation maximale des buffers et donc sur la taille des buffers à utiliser. Nous calculons à cet effet *l'écart maximum relatif* entre les deux graphiques. Les résultats obtenus sont repris au tableau 4-14.

	Ecart maximum relatif				
	Ligne 1	*Ligne 2*	*Ligne 3*	*Ligne 4*	*Ligne 5*
CRC 16	55,5	37,5	44,5	62	52,5
INTERNET CHECKSUM	232	104,5	28	25	68
XOR SOURCE DEST					
XOR DEST	103,5	54	57	29	38,5
DEST	38	43	132	21	20

Tableau 4-14 : Ecart maximum relatif entre les graphiques d'occupation instantanée des buffers TCPDUMP et NETFLOW

L'erreur dans le "pire des cas instantanés" est importante. *L'écart maximum relatif* entre les occupations instantanées des buffers TSH et NETFLOW est en moyenne de +/- 62 %. Deux valeurs obtenues dépassent même les 100 %. *L'erreur d'occupation des buffers est donc très importante*. Il nous paraît donc difficile d'effectuer des *prévisions* concernant l'occupation maximale des buffers des lignes de sortie sur base d'une trace NETFLOW fournie par un routeur.

4.4 Conclusions

Nous avons montré à la section 4.1.3 que la distribution du volume de trafic utilisé était peu uniforme. Nous avons également montré à la section 4.1.4 que la distribution des flux en fonction de leur longueur était très inhomogène. Nous avons donc utilisé de "mauvaises" traces du point de vue de l'homogénéité.

Nous obtenons malgré tout d'**excellents résultats en ce qui concerne la répartition *globale* du trafic sur les différentes lignes de sortie**, du moins en ce qui concerne les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE-DESTINATION et XOR DESTINATION. **L'utilisation des méthodes de hachage pour effectuer du LOAD BALANCING s'avère donc potentiellement intéressante**, en tous cas au niveau d'un routeur pris en particulier (nous n'avons pas effectué d'études sur un réseau complet et ne pouvons donc affirmer avec certitude que l'utilisation du LOAD BALANCING s'avère intéressante à cette échelle). Les résultats obtenus ne permettent pas de réellement départager les méthodes CRC16, INTERNET CHECKSUM, XOR SOURCE-DESTINATION et XOR DESTINATION. Ils sont fort proches et pourraient être différents pour d'autres traces de trafic. **Il serait intéressant de pouvoir effectuer des mesures sur des temps plus longs et sur des traces *réelles* (non anonymisées).**

Les résultats obtenus sont nettement moins "bons" en ce qui concerne la répartition instantanée du trafic sur chacune des lignes de sortie. Nous avons montré à la section 2.1 qu'ils étaient dus à l'existence d'un petit nombre de "gros" flux.

Nous avons obtenu une "mauvaise" occupation des buffers au cours du temps et d'une ligne de sortie à l'autre. Elle est due à la "mauvaise" répartition instantanée du trafic. Il semble **difficile d'évaluer les performances des différentes méthodes de répartition de trafic sur base du critère "occupation des buffers"**. Nous avons constaté que l'occupation maximale des buffers des différentes lignes de sortie était du même ordre de grandeur que la taille du buffer de la ligne d'entrée, ce qui nous a amené à penser que **le LOAD BALANCING n'est pas en elle-même une technique très efficace pour minimiser l'occupation des buffers**.

Un autre aspect de l'étude expérimentale portait sur la **comparaison des résultats obtenus après application des méthodes de hachage sur une trace de format TCPDUMP et les résumés de flux NETFLOW.** Rappelons que l'intérêt des résumés de flux NETFLOW réside dans le fait qu'ils permettent d'évaluer les méthodes de répartition de trafic sur des intervalles de temps relativement grands, de l'ordre de plusieurs heures ou plusieurs jours (cfr. chapitre 3). Ces résumés de flux NETFLOW se font bien entendu au détriment de la précision et donc de la finesse de l'analyse. Nos résultats expérimentaux ont montré que NETFLOW approxime relativement bien TCPDUMP en ce qui concerne la répartition du trafic sur les différentes lignes de sortie après application des méthodes de LOAD BALANCING. **IL paraît donc raisonnable d'envisager l'utilisation de NETFLOW à la place de TCPDUMP pour mesurer la répartition instantanée du trafic sur les différentes lignes de sortie durant des intervalles de temps relativement longs. Nous ne pouvons cependant pas envisager l'utilisation de NETFLOW pour mesurer l'occupation instantanée des buffers** des différentes lignes de sortie. L'erreur dans le "pire des cas instantanés" est effectivement trop importante. Il paraît difficile d'effectuer des *prévisions* concernant l'occupation maximale des buffers des lignes de sortie sur base d'une trace NETFLOW fournie par un routeur.

Chapitre 5

Conclusions et extensions futures

5.1 Conclusions

Nous avons expliqué au chapitre 1 pourquoi il est intéressant d'effectuer du LOAD BALANCING dans les gros réseaux IP INTERNET. Nous avons vu qu'une bonne méthode de répartition de trafic devait reposer sur du code *simple*, étant donné les débits élevés auxquels fonctionnent les routeurs aujourd'hui (on arrive au Terabits/s).

Nous avons expliqué au chapitre 2 qu'une bonne méthode de répartition de trafic devait respecter l'ordre dans lequel les paquets arrivent à destination, la majorité du trafic actuel sur Internet étant du trafic TCP [THOM98]. Nous avons vu que, parmi les différentes méthodes de répartition de trafic, certaines dites *méthodes de hachages* s'avèrent être, en théorie, fort intéressantes.

Nous avons essayé de vérifier au chapitre 4 si ceci était également vrai en pratique, c'est-à-dire dans les gros réseaux IP INTERNET. Nous avons utilisé une trace correspondant au trafic capté pendant un intervalle de temps de +/- 90 secondes sur une ligne d'accès INTERNET d'une grosse université américaine.

Nous avons obtenu de bons résultats en ce qui concerne la répartition *globale* du trafic sur les différentes lignes de sortie, du moins en ce qui concerne les méthodes de hachage CRC16, INTERNET CHECKSUM, XOR SOURCE et XOR DESTINATION. L'utilisation des méthodes de hachage pour effectuer du LOAD BALANCING s'avère donc potentiellement intéressante, en tous cas au niveau d'un routeur pris en particulier (nous n'avons pas effectué d'études sur un réseau complet et ne pouvons donc affirmer avec certitude que l'utilisation du LOAD BALANCING s'avère intéressante à cette échelle). Les résultats obtenus n'ont pas permis de réellement départager les méthodes CRC16, INTERNET CHECKSUM, XOR SOURCE et XOR DESTINATION. Ils sont fort proches et pourraient être différents pour d'autres traces de trafic.

Les résultats obtenus sont nettement moins "bons" en ce qui concerne la répartition *instantanée* du trafic sur chacune des lignes de sortie. Nous avons montré à la section 2.1 qu'ils étaient dus à l'existence d'un petit nombre de "gros" flux.

L'occupation des buffers variant fortement au cours du temps et d'une ligne de sortie à l'autre, il nous a été difficile d'évaluer les performances des différentes méthodes de répartition de trafic sur base de ce critère. Nous avons de plus constaté une occupation maximale des buffers des différentes lignes de sortie semblable à l'occupation maximale du buffer de la ligne d'entrée, ce qui nous a amené à penser que *le* LOAD BALANCING *n'est pas en elle-même une technique très efficace pour minimiser l'occupation des buffers.*

La trace utilisée (+/- 90 secondes) ne nous permet pas de conclure avec certitude à l'efficacité des méthodes de hachage sur des intervalles de temps relativement longs.

L'intervalle de temps relativement court de la trace utilisée est dû au logiciel de capture utilisé. Il s'agit d'un logiciel de capture stockant les informations *paquet par paquet*. Le choix d'un tel logiciel est logique parce qu'il est susceptible de fournir des résultats extrêmement fiables. Il nécessite cependant le stockage d'un volume d'informations considérable, ce qui oblige dès lors à effectuer les mesures expérimentales sur des temps relativement courts.

Si on veut comparer les différentes méthodes de hachage *sur des intervalles de temps plus longs*, il faut utiliser un logiciel de capture ne stockant pas les informations *paquet par paquet* mais les résumant en mémoire. Une manière de procéder est de résumer les informations *flux par flux*. Le risque est cependant une importante perte de précision et de charger le CPU du routeur qui exporte l'information NETFLOW, comme expliqué au chapitre 3. Dans le but de mesurer la perte de précision, nous avons, à partir de notre trace de départ, créé une trace stockant les informations *flux par flux*. Nos différents algorithmes de hachage ont alors été appliqués sur cette nouvelle trace. Les résultats obtenus montrent une répartition de trafic semblable à celle obtenue avec la trace stockant les informations paquet par paquet. *L'utilisation de traces stockant les informations flux par flux peut donc être envisagée en ce qui concerne la mesure de la répartition instantanée du trafic.* Elle ne semble cependant pas pouvoir être envisagée en ce qui concerne la *mesure de l'occupation instantanée des buffers* des différentes lignes de sortie. Il paraît en tous cas difficile d'effectuer des *prévisions* concernant l'occupation maximale des buffers des lignes de sortie sur base d'une trace fournissant des informations flux par flux.

5.2 Extensions futures possibles du mémoire

5.2.1 Les résultats obtenus montrent que ce ne sont pas des flux les plus nombreux dont il faut tenir compte. Ce sont surtout des "gros" flux (de longueur ≥ à 10^6 bytes) dont il faut tenir compte. Ce sont eux qui représentent l'essentiel (66%) du volume. C'est surtout pour eux que le LOAD BALANCING semble utile. **On pourrait dès lors imaginer envoyer les paquets appartenant aux "petits" flux sur une ou plusieurs lignes de sortie déterminées** (par exemple, les lignes de sortie 1 et 2) **à l'aide d'un algorithme de hachage simple** (par ex: IP DESTINATION) **et de n'appliquer les algorithmes de hachage complexes** (par exemple: CRC16) **que sur les paquets appartenant aux flux supérieurs à un certain volume**. Cette solution n'est cependant pas à retenir, aucune économie significative en terme de CPU n'étant à espérer. Les algorithmes de hachage complexes peuvent en effet être exécutés en parallèle avec l'algorithme de routage. L'économie réalisée en terme de hachage complexe n'est de plus que de +/- 34 % en ce qui concerne notre trace. La solution consistant à n'appliquer les algorithmes de hachage complexe que sur les paquets appartenant aux gros flux ne semble donc pas très intéressante dans le cadre de cette étude.

5.2.2 **Un LOAD BALANCING effectué de manière *dynamique* devrait pouvoir améliorer la répartition *instantanée* du trafic sur les différentes lignes de sortie.**

Pour être efficace, le LOAD BALANCING dynamique ne peut être effectué que durant des **intervalles de temps relativement courts** (quelques secondes maximum) et ceci de manière à pouvoir suivre les variations très rapides des volumes instantanés de trafic sur les différentes lignes de sortie. Ceci peut évidemment poser problème: **la dynamique de l'adaptation du LOAD BALANCING risque d'engendrer des instabilités à l'échelle globale du réseau. Comment limiter les instabilités?**

Il devrait peut-être être possible de limiter en grosse partie ces instabilités en utilisant l'extension OMP [VIL99]. Cette extension est utilisable au sein de OSPFv2 [MOY91]. OMP (OPTIMIZED MULTIPATH) étant basé sur ECM (EQUAL COST MULTIPATH) [HOP00], nous commençons par décrire ce dernier.

ECM a pour objectif de répartir de manière *équitable* le trafic correspondant à une entrée spécifique de la table de routage pour laquelle on aurait plusieurs chemins minimums de coûts égaux. La technique ECM se base sur la *valeur* produite par l'algorithme de hachage. L'ensemble des valeurs possibles fournies par l'algorithme de hachage est divisé en régions contiguës d'égales grandeurs. En cas de changement du nombre de routes possibles pour une destination particulière, l'espace de valeur est redistribué de manière équitable entre le nombre réduit ou augmenté de lignes de sortie. Un exemple est montré à la figure 5.1.

Valeur binaire fournie par l'algorithme de hachage: *O1234567 O1234567 O1234567 O1234567 O1234567*

1	2	3	4	5
1	2		4	5

O1234567 O1234567 O1234567 O1234567 O1234567

Figure 5.1 : EQUAL COST MULTIPATH

Dans cette figure, la ligne 3 (liée à la région 3) est supposée disparaître. L'espace de valeur se redistribue alors de manière équitable entre les régions 1, 2, 4 et 5 restantes.
Nous pouvons observer que ¼ de la région 2 est devenue région 1, que ½ de la région 3 est devenue région 2, que ½ de la région 3 est devenue région 4 et que ¼ de la région 4 est devenue région 5. Puisque chaque région originale représente 1/5 des flux, la rupture de séquence totale des flux affecte 1/5*(1/4+1/2+1/2+1/4) = 3/10 du nombre de flux.

ECM provoque donc une *rupture de séquence* pour un nombre non négligeable de flux TCP au moment du changement du nombre de lignes de sortie [HOP00]. Ceci peut entraîner un changement dans l'ordre d'arrivée des paquets à la machine destination. Nous avons expliqué, à la section 2.2.2, l'impact négatif que ceci pouvait avoir au niveau du mécanisme de contrôle de congestion TCP.

ECM pose d'autres problèmes :

1) Il ne permet d'effectuer du LOAD BALANCING qu'au niveau *local*. LE LOAD BALANCING devrait cependant pouvoir être effectué au niveau *global*, c'est à dire à l'échelle de tout un *domaine*[38] : l'optimisation de la répartition du trafic nécessite efectivement de tenir compte des variations de charge dans *tout* le domaine.
2) Le trafic est réparti de manière *égale* entre les différentes lignes de sortie. Or, si on veut tenir compte des variations de charge, il faut répartir le trafic de manière *inégale* sur les différentes lignes de sortie.

C'est ici qu'intervient OMP [VIL99]. L'apport de OMP se situe principalement à deux niveaux :

1) IL utilise des LSA[39] opaques[40], ce qui lui permet de distribuer l'information concernant la *charge* des différentes lignes.

[38] Ensemble des routeurs d'un système autonome qui partagent le même protocole de routage (OSPF ou ISIS par exemples)

[39] Messages d'état de liaison (LINK STATE ADVERTISEMENT). Chaque routeur est responsable de la description de sa partie locale du domaine via des messages d'état de liaison. Ces messages sont régulièrement envoyés vers les autres routeurs à l'aide d'un protocole particulier.

[40] Le *opaque* LSA [VIL99 p.19-21] [COL98] est un nouveau type de LSA dont le but premier est de fournir un support aux extensions futures de OSPF. L'information contenue dans le *opaque* LSA peut être directement utilisée par OSPF ou indirectement par certaines applications qui veulent distribuer de l'information partout dans le domaine OSPF.

2) Il permet un ajustage *incrémental* de la charge en même temps qu'une vitesse d'ajustement relativement rapide. L'ajustage *incrémental* permet de minimiser les problèmes d'oscillation liés à la stabilité de routage et à la variation de charge. Ces problèmes d'oscillation peuvent effectivement conduire à une "mauvaise" répartition de la charge, c'est à dire à un "mauvais" LOAD BALANCING. Le fonctionnement de OMP dépasse le cadre de ce mémoire et est décrit dans [VIL99].

5.2.3 Le LOAD BALANCING dynamique pourrait tenir compte de la répartition instantanée des *gros* flux sur les différentes lignes de sortie. En cas de congestion d'une ligne de sortie, il pourrait rediriger préférentiellement certains paquets plutôt que d'autres.

Seule la connaissance de la répartition instantanée des " gros" flux (ceux de longueur ≥ à 10^6 bytes) est effectivement nécessaire, comme montré à la figure 5.2.

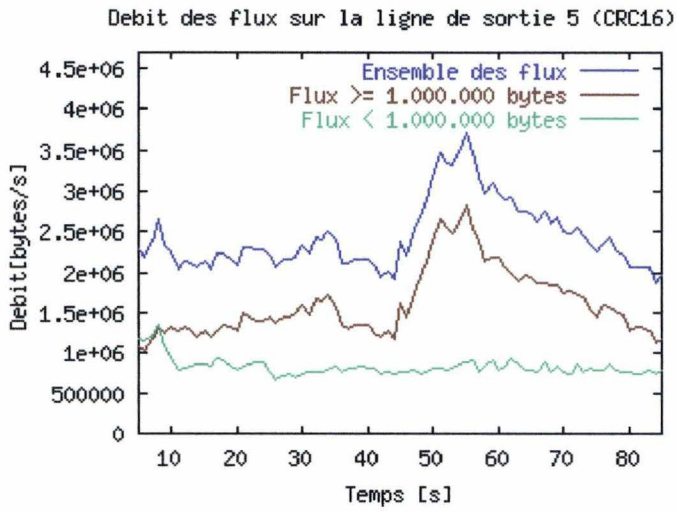


Figure 5.2:Débit instantané de l'ensemble des flux, des flux ≥ 10^6 bytes et des flux < 10^6 bytes sur la ligne de sortie 5 après hachage CRC16.

Le trafic correspondant aux flux inférieurs à 10^6 bytes est relativement constant. Nous pouvons constater une évolution semblable des flux de longueur ≥ à 10^6 bytes et de l'ensemble des flux. Le LOAD BALANCING dynamique pourrait donc uniquement tenir compte de la répartition instantanée des *gros* flux sur les différentes lignes de sortie.

On pourrait imaginer de procéder de la manière suivante. Lorsqu'aucune ligne de sortie n'a le débit instantané de ses gros flux supérieur à une certaine valeur (y par exemple), les paquets sont répartis normalement sur *toutes* les lignes de sortie à l'aide d'un des algorithmes de hachage étudiés au chapitre 4. Lorsque, par contre, le débit instantané des gros flux d'une ligne de sortie x dépasse la valeur y, cette ligne de sortie est supposée entrer en congestion. Le routeur décide alors de ne plus envoyer *un certain nombre de paquets* vers cette ligne de sortie x, du moins jusqu'au moment où le débit instantané des gros flux de la ligne de sortie x redescend en dessous de la valeur y. L'algorithme de hachage s'applique dès lors, pour ces

paquets, sur les lignes de sortie non encombrées. Tout le problème consiste bien entendu à déterminer la valeur optimale de *y* ainsi que les paquets devant être redirigés.

Il est difficile de choisir une valeur optimale pour ***y*** à partir du moment où on ne connaît pas la taille des buffers des lignes de sortie. La valeur de ***y*** ne peut pas être choisie trop élevée sous peine de supprimer tout l'aspect "dynamique" du LOAD BALANCING. Un compromis doit donc s'instaurer entre le nombre de reséquencements acceptables et la "dynamique" du LOAD BALANCING. En ce qui concerne notre trace, une valeur comprise entre 2,5 et 3 10^6 semble réaliste (figures 4-9 à 4.12). Nous ne pouvons cependant affirmer que la valeur optimale de *y* se trouverait dans le même intervalle si nous avions capturé notre trace à un moment différent ou si nous avions capturé notre trace dans un autre réseau INTERNET BACKBONE. La valeur optimale de *y* pourrait effectivement être très différente d'un moment à l'autre et/ou d'un réseau à l'autre. Il serait peut-être possible, en effectuant un grand nombre de mesures statistiques sur les différents types de trafic possibles dans les réseaux INTERNET BACKBONE, de trouver une *liste* de valeurs optimales pour *y*. Chaque valeur optimale dépendrait du moment et/ou du type de trafic présent dans le réseau BACKBONE. Nous ne pouvons bien entendu pas affirmer qu'une telle liste existe vraiment, ou même, si elle existe, qu'il est pratiquement possible de la trouver.

Le nombre de paquets à rediriger ne peut pas être égal à l'ensemble des paquets devant normalement se diriger vers la ligne de sortie *x*. Ceci reviendrait effectivement à ne plus envoyer aucun trafic vers la ligne de sortie *x*.
Il semble réaliste de ridiriger les paquets appartenant aux gros flux. Intuitivement, c'est en "cassant" les gros flux que l'on devrait pouvoir obtenir le minimum de déséquencements[41]. Pour transférer le même volume de bytes d'une ligne de sortie à l'autre, il faut effectivement "casser" beaucoup moins de "gros" flux que de "petits".

Les paquets appartenant aux gros flux ne peuvent cependant être redirigés que dans la mesure où les routeurs sont capables de les repérer. *Comment un routeur peut-il repérer les paquets appartenant à un gros flux ?*
Grâce au cache NETFLOW. NETFLOW maintient effectivement dans le cache du routeur une table des derniers flux arrivés dans le routeur. Chaque ligne de cette table caractérise un flux et comprend l'adresse IP source, l'adresse IP destination, le port source, le port destination et le protocole du flux. Il est parfaitement possible de rajouter une colonne à cette table si on le souhaite. On peut ainsi rajouter une colonne comptabilisant, pour chaque flux, le nombre de bytes déjà arrivés dans le routeur. A chaque arrivée d'un paquet IP, le routeur peut alors "matcher" la table NETFLOW suivant l'adresse IP source et l'adresse IP destination du paquet et déterminer s'il appartient à un gros flux ($\geq 10^6$ bytes) ou non.

On peut également se poser la question suivante. *Faut-il "casser" tous les gros flux ou seulement les "nouveaux" gros flux, c'est-à-dire ceux dont le nombre de bytes comptabilisés dans le cache du routeur va dépasser10^6 bytes durant la période de saturation de la ligne x?*
Il vaut mieux éviter de rediriger *tous* les paquets appartenant aux *gros* flux. Ceci provoquerait effectivement une sous utilisation de la ligne *x* puisqu'elle ne contiendrait plus que les paquets appartenant aux petits flux, c'est à dire seulement +/-34% du volume *moyen* de la ligne de sortie.

---

[41] Nous avons expliqué, à la section 2.2.2, l'impact négatif que les déséquencements pouvaient avoir au niveau de mécanisme de contrôle de congestion TCP.

Une solution peut consister à ne rediriger, du moins *dans un premier temps, que* les "nouveaux flux", c'est à dire ceux dont le nombre de bytes comptabilisés dans le cache du routeur ne dépasse pas 10^6 bytes au début de la saturation de la ligne *x* mais *va* les dépasser durant la période de saturation de la ligne *x*. Les flux de longueur déjà supérieure à 10^6 bytes *au début* de l'encombrement de la ligne de sortie *x* sont donc maintenus sur la ligne de sortie *x*. Si ça ne suffit pas, c'est-à-dire si le trafic continue à augmenter sur la ligne saturée, alors il faudra également " casser" une partie des flux dépassant déjà les 10^6 bytes au début de la période de saturation de la ligne. Le problème est bien entendu de savoir lesquels. Une solution serait de procéder de manière probabiliste. Pour ce faire, il faudrait pouvoir déterminer statistiquement quels gros flux il vaut mieux "casser". Vaut-il mieux "casser" les flux de longueurs comprises entre 10^6 et 10^7 bytes (par exemple) ou vaut-il mieux "casser" les flux de longueurs comprises entre 10^8 et 10^9 bytes (par exemple)? La réponse à cette question n'est pas évidente à priori. Pour y répondre, il faudrait effectuer un grand nombre de mesures statistiques sur divers types de trafic. Il faudrait pouvoir vérifier l'impact de la distribution des longueurs des flux sur l'optimalité des algorithmes de LOAD BALANCING. Certaines distributions de longueur de flux pourraient effectivement rendre optimales certaines stratégies de LOAD BALANCING et pas d'autres. On pourrait également utiliser un simulateur (le simulateur NS [NS95] par exemple) pour générer des flux de longueurs variables. Tout le problème ici, est de *produire un système stable avec du trafic instable.*

Enfin, une dernière question. *Que se passe-t-il pour les gros flux si toutes les lignes de sortie sont encombrées ?*
Ceci ne se produira pas si la valeur de ***y*** est choisie suffisamment grande, c'est-à-dire au moins égale au débit *maximal* de la ligne d'entrée divisé par le nombre de lignes de sortie (encombrées ou non).

5.2.4 Le LOAD BALANCING devrait également, idéalement, minimiser l'occupation des buffers des différentes lignes de sortie.

Nous avons vu à la section 4.2.2.3 que l'occupation maximale des buffers des différentes lignes de sortie était du même ordre de grandeur que l'occupation maximale du buffer de la ligne d'entrée. Le LOAD BALANCING ne semble donc pas en elle-même une technique très efficace pour minimiser l'occupation des buffers. On peut imaginer résoudre le problème en adaptant *dynamiquement* l'occupation des buffers en s'appuyant sur les données recueillies par un logiciel de capture de trafic présent sur les différentes lignes de sortie. On peut par exemple imaginer utiliser un mécanisme de type RED[42] basé sur le *taux d'occupation* des buffers. Supposons par exemple une ligne de sortie *x,* dont le taux d'occupation du buffer dépasse une certaine valeur maximale appelée *seuil maximum*. Les *prochains gros flux* devant normalement être envoyés vers cette ligne *x* pourraient être redirigés vers une ou plusieurs autres lignes de sortie. La redirection ne s'effectuerait que vers des lignes de sortie ayant un taux d'occupation de leur buffer relativement faible, inférieur à une certaine valeur appelée *seuil minimal*, de manière à ne pas saturer, à leur tour, les buffers des nouvelles lignes de sortie.

Nous pensons toutefois que l'adaptation dynamique de l'occupation des buffers risque d'introduire une complexité supplémentaire dans les routeurs internes. Ce n'est pas exactement ce que l'on souhaite. Le LOAD BALANCING dynamique n'a de toute façon pas en

[42]RANDOM EARLY DETECTION [FLO93]

lui-même pour objectif principal d'optimiser l'occupation des buffers. La principale raison d'effectuer du LOAD BALANCING dynamique se situe surtout au niveau de l'*amélioration des répartitions instantanées du trafic* sur les différentes lignes de sortie sélectionnées par l'algorithme de routage..

Annexe 1

Bibliographie

Introduction

[ALC00] LG, *Alcatel câble: des fibres ultra-rapides en provenance des Flandres françaises*, Network&Telecom, Mai 2000

[BAL96] K. Bala, F.R.K Chung, C.A. Brackett, *Optical wavelength routing, translation and packet/cell switched networks*, IEEE/OSA J. Lightware Technology, 14(3):336-343, 1996

[MAL98] G. Malkin, *RIP Version 2*, RFC 2453, Novembre 1998

[MOY91] J. Moy, *OSPF Version 2*, RFC 2178, Juillet 1997

[ORA90] D. Oran, *OSI IS-IS Intra-domain Routing Protocol*, RFC 1142, Février 1990

[POS81-b] Jon Postel, *Transmission Control Protocol*, Septembre 1981

1. LOAD BALANCING dans les gros réseaux INTERNET IP

[ATT00] *Carte du réseau Backbone* de *AT&T IP, AT&T*, 2001, disponible à l'adresse http://www.ipservices.att.com/backbone/images/bbone_2001.pdf

[FRO90] Christine Froidevaux, Marie-Claude Gaudel, Michèle Soria, *Types de données et algorithmes,* Mc Graw-Hill, 1990

[HOP00] C. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, RFC 2992, Novembre 2000.

[MAL98] G. Malkin, *RIP Version 2*, RFC 2453, Novembre 1998

[MOY89] J.Moy, *OSPF specification*, RFC 1131, Octobre 1989

[MOY91] J. Moy, *OSPF Version 2*, RFC 2178, Juillet 1997

[ORA90] D. Oran, *OSI IS-IS Intra-domain Routing Protocol*, RFC 1142, Février 1990

[PER92] Perlman, R 1992. *Interconnections: Bridges and Routers.* Addison-Wesley, Mars 1992

[THA00] Thaler, *Multipath Issues and Multicast Next-Hop Selection,* Novembre 2000

[vBNS99] *Carte du réseau Backbone* de *vBNS Backbone Network Map*, MCIWORDCOM, 1999, disponible à l'adresse http://www.vbns.net/netmaps/backbone.html

[VIL99] Curtis Villamizar, *OSPF Optimized Multipath (OSPF-OMP)*, draft-ietf-ospf-omp-02, Février 1999

2. Les Méthodes de distribution du trafic

[BRA94] Lawrence Brakmo, Sean O'Malley, Larry Peterson, *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, in ACM SIGCOMM 94, 1994

[BRO61] W. W Peterson, D.T Brown, *Cyclic codes for error detection*, in Proc. IRE, vol. 49, 1961

[CLA00] Sean McCreary, K. Claffy, *Trends in Wide Area IP Traffic Patterns*, Caida, 2000

[DEE89] S. Deering, *Host Extensions for IP multicastin*", RFC 1112, Août 1989

[DEE97] S. Deering, D. Estrin, D. Farinacci, Van Jacobson, D. Meyer, L. Wei, *Protocol Independent Multicast version 2 Dense Mode Specification*, Internet draft, draft-ietf-idmr-pim-dm-06.txt, 1997

[FLE83] J. Fletcher, *An arithmetic checksum for serial transmissions*, IEEE Trans. Comm., vol. Comm-30, pp. 247-252, Janvier 1983

[GREE92] D. Greene, B. Lyles, *Reliability of adaptation layers*, Protocols for High-Speed Networks III, Proc. IFIP 6.1/6.4 Workshop, B. Pehrson. P. Gunningberg, and S. Pink, Eds., 1992

[JAC90]Van Jacobson, *Berkeley TCP Evolution From 4.3-Reno*, in Proceedings of the British Columbia Internet Engineering Task Force, Juillet 1990

[JAI92] Raj Jain, *A comparaison of Hashing schemes for Address Lookup in Computer Networks*, IEE Transactions on Communications, Octobre 1992.

[MIL98] *The nature of the beast: recent traffic measurements from an Internet Backbone*, Greg Miller and Kevin Thompson, kc@caida.org, Avril 1998

[PAR95] C. Partridge, J. Hughes, J. Stone, *Performance of Checksums and CRCs over Real Data*, ACM Sigcomm'95, 1995

[POS81-a] Jon Postel, *Internet Protocol*, RFC 791, Septembre 1981

[POS81-b] Jon Postel, *Transmission Control Protocol*, Septembre 1981

[SHR94] M. Shreedhar, G. Varghese. *Efficient Fair Queuing by Deficit Round Robin*, Technical Report WU94-17, Washington University, 1994.

[STO 98] Jonathan Stone, Michael Greenwald, *Performance of Chechsums and CRC's over Real Data*, IEEE/ACM Transactions on Networking, vol. 6, n°5, Octobre 1998

[THA98] David Thaler et Chinya Ravishankar, *Using name-based Mappings to Increase Hit Rates*, IEEE/ACM, Transactions on Networking, Février 1998

[THOM98] K. Thompson, G. Miller, *The nature of the beast: recent traffic measurements from an Internet Backbone*, INET 98, Avril 1998

3. Simulation du trafic INTERNET BACKBONE

[CFL01] Caida. *Cflowd: Traffic Flow Analysis Tool*, disponible à partir du site http://www.caida.org/tools/measurement/cflowd, 2001

[CIS00-a] *White Paper NetFlow Services and Applications*, Juin 2000, disponible à partir du site http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm

[CIS00-b] *Catalyst 5000 Family NetFlow Feature Card II* , Cisco, Juillet 2000, disponible à partir du site http://www.cisco.com/warp/public/cc/pd/si/casi/ca5000/prodlit/nffc2_ds.htm

[CIS00-c] *White Paper NetFlow Services and Applications, Aggregation Schemes, Detail Host Matrix*, Juin 2000, disponible à partir du site http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm

[CIS99] *Netflow Export Datagram Format*, Cisco, Juillet 1999, disponible à partir du site http://www.cisco.com/univercd/cc/td/doc/product/rtmgmt/nfc/nfc_3_0/nfc_ug/nfcform.htm

[DUM01] *Tcpdump/lipcap*, disponible à partir du site http://www.tcpdump.org, 2001

[JAC92] Van Jacobson, Craig Leres, Steve Mc Canne, Université de Californie, Berkeley, Juin 1992

[LEI99] Simon Leinen , *Flow-Based Monitoring and Analysis*, SWITCH, 10 Septembre 1999, http://www.switch.ch/tf-tant/floma/frankfurt/floma.html

[NET99] Cisco. *Netflow Services and Applications*, White Paper, disponible à partir du site http://www.cisco.com/warp/public/732/netflow, 1999

[PLON00] Mark Fullmer, Steve Romig, *The OSU Flow-tools Package and Cisco Netflow Logs*, The Ohio State University, Usenix, 14th Systems Administration Conference, Décembre 2000

[POS80] Jon Postel, *User Datagram Protocol*, RFC 768, Août 1980

[POS81-b] Jon Postel, *Transmission Control Protocol*, Septembre 1981

[ROM00] Dave Plonka, *A Network Traffic Flow Reporting and Visualization Tool*, University of Wisconsin-Madison, Usenix, 14th Systems Administration Conference, Décembre 2000

4. Etude expérimentale

[BRA89] R. Braden, *Requirements for Internet Hosts – Communication Layers*, RFC 1122, Octobre 1989

[MO90] J. Mogul, S. Deering, *Path MTU Discovery*, RFC 1191, Novembre 1990

[NLA1] IL s'agit de l'université d'état du COLORADO. La trace utilisée a été crée le 11 Octobre 2000 pendant un intervalle de temps de +/- 90 secondes sur la ligne d'accès INTERNET de l'université. Elle a été créée par l'équipe MOAT (MEASUREMENT & OPERATIONS ANALYSIS TEAM) de NLANR (NATIONAL LABORATORY FOR APPLIED NETWOK RESEARCH). On peut la télécharger à partir du site de NLANR (ftp://moat.nlanr.net/pub/MOAT/Traces). Il s'agit d'un fichier anonymisé *cos-971136250-1.tsh.enc.gz.*

[PER99] Puqi Perry Tang, Tsung-Yuan Charles Tai, *Network Traffic Characterization Using Token Bucket Model*, Proceedings of the conference on Computer Communications, IEEE Infocom, Mars 1999

[THOM98] K. Thompson, G. Miller, *The nature of the beast: recent traffic measurements from an Internet Backbone*, INET 98, Avril 1998

5. Conclusions et extensions futures

[AWD99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, *Requirements for Traffic Engineering over MPLS*, RFC 2702, Septembre 1999

[COL 98] R. Coltun, *The OSPF Opaque LSA Option*, RFC 2370, Juillet 1998

[FLO93] S. Floyd, V.Jacobson, *Random Early Detection gateways for congestion avoidance*, IEEE/ACM Trans. Networking, V1, N4, pp. 397-413, 1993

[HOP00] C. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, RFC 2992, Novembre 2000.

[MOY91] J. Moy, *OSPF Version 2*, RFC 2178, Juillet 1997

[NS95] DARPA (DEFENSE ADVANCED RESEARCH PROJECTS), USC/TSI, XEROX PARC, UCB, 1995. La documentation concernant ce simulateur est disponible à l'adresse http://www.isi.edu/nsnam/ns/ns-documentation.html

[THOM98] K. Thompson, G. Miller, *The nature of the beast: recent traffic measurements from an Internet Backbone*, INET 98, Avril 1998

[VIL99] Curtis Villamizar, *OSPF Optimized Multipath (OSPF-OMP)*, draft-ietf-ospf-omp-02, Février 1999

Annexe 2

Implémentation du LOAD BALANCING

Une batterie d'algorithmes a été réalisée dans le cadre de cette étude. Le schéma logique correspondant est indiqué ci-dessous:

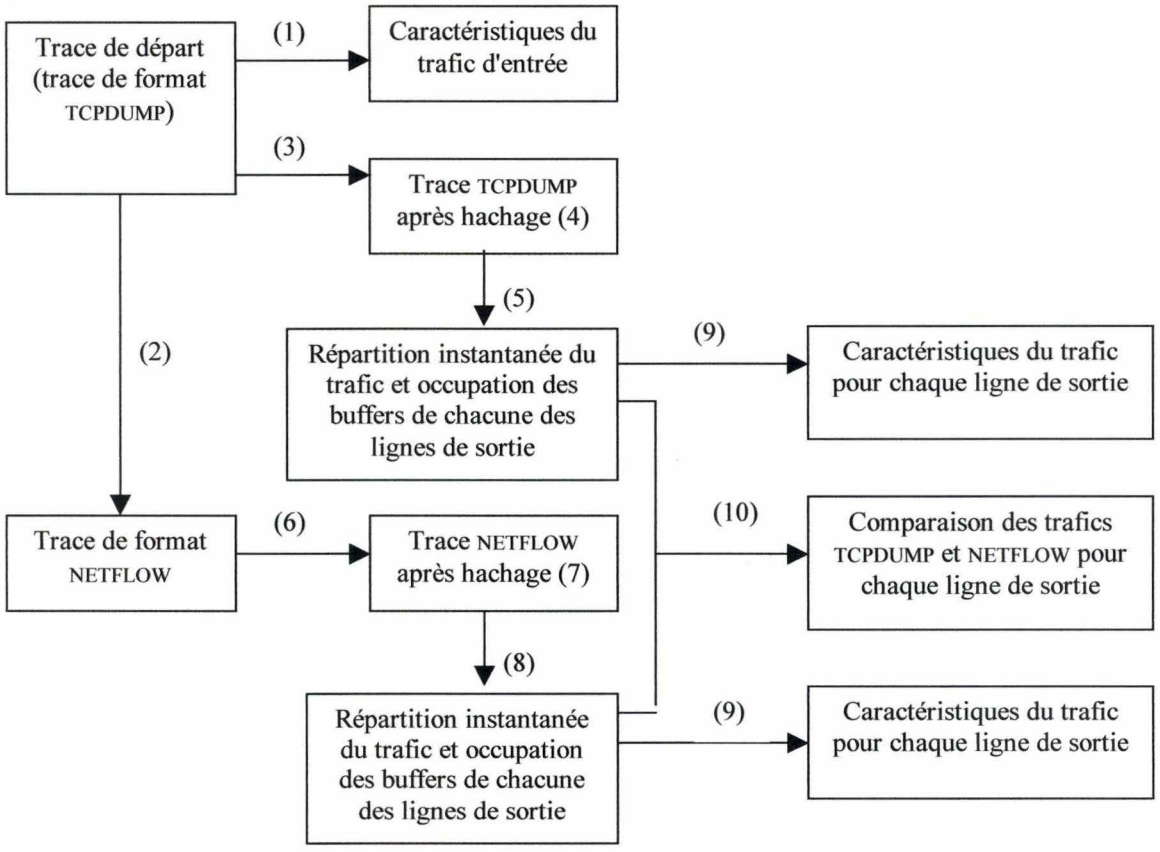


Schéma logique des algorithmes réalisés

(1) Ensemble d'algorithmes testant les caractéristiques de la trace TCPDUMP utilisée (détermination de la répartition instantanée du trafic, de la répartition instantanée des flux, de l'occupation instantanée du buffer de la ligne d'entrée du routeur, de la distribution des paquets et des rafales, de la distribution du volume de trafic et des flux en fonction des adresses IP, de la distribution des flux en fonction de leur longueur, ...).

(2) Algorithme de transformation de la trace de format TCPDUMP (informations stockées *paquet par paquet*) en une trace de format NETFLOW (informations stockées *flux par flux*)

(3) Algorithme appliquant une méthode de hachage choisie par l'utilisateur sur chacun des *paquets* de la trace TCPDUMP de départ. Le nombre de lignes de sortie est également choisi par l'utisateur.

(4) La trace TCPDUMP contient à présent, pour chaque *paquet*, le numéro de la ligne de sortie calculé par l'algorithme de hachage.

(5) Algorithme simulant la répartition instantanée du trafic et l'occupation instantanée des buffers sur chaque ligne de sortie à partir de la trace TCPDUMP modifiée.

(6) Algorithme appliquant une méthode de hachage choisie par l'utilisateur sur chacun des *flux* de la trace NETFLOW de départ. Le nombre de lignes de sortie est également choisi par l'utisateur.

(7) La trace NETFLOW contient à présent, pour chaque *flux*, le numéro de la ligne de sortie calculé par l'algorithme de hachage.

(8) Algorithme simulant la répartition instantanée du trafic et l'occupation instantanée des buffers sur chaque ligne de sortie à partir de la trace NETFLOW modifiée.

(9) Ensemble d'algorithmes testant les caractéristiques du trafic présent sur les différentes lignes de sortie (répartition des flux sur les différentes lignes de sortie, du nombre de flux, du nombre de "gros" flux, de la répartition instantanée des "gros" flux,...)

(10) Ensemble d'algorithmes permettant de comparer les répartitons instantanées de trafic obtenues et les occupations instantanées des buffers obtenues après application des méthodes de hachage sur les traces TCPDUMP et NETFLOW. L'*erreur* commise en utilisant NETFLOW à la place de TCPDUMP est également calculée.